МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет електроніки
Кафедра промислової електроніки

**О.О. АБАКУМОВА, Д.А. МИКОЛАЄЦЬ**

# МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ КОМП'ЮТЕРНИХ ПРАКТИКУМІВ З ДИСЦИПЛІНИ «ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

для студентів спеціальності 171 Електроніка
спеціалізації Електронні системи

Київ 2017

*Гриф надано Вченою радою факультету електроніки КПІ ім. Ігоря Сікорського*
*(Протокол № 05/17 від 29 травня 2017р.)*

Рецензент: *В. А. Казміренко*, канд. техн. наук, доцент,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Відповідальний
редактор: *Т. А. Хижняк,* канд. техн. наук, доцент,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Методичні вказівки до виконання комп'ютерних практикумів з дисципліни «Об'єктно-орієнтоване програмування» для студентів спеціальності 171 Електроніка спеціалізації Електронні системи / Уклад.: Абакумова О.О., Миколаєць Д.А. – К. : КПІ ім. Ігоря Сікорського, 2017. – 84 с.

Методичні вказівки містять роз'яснення щодо виконання 7 комп'ютерних практикумів, передбачених робочою навчальною програмою кредитного модуля «Об'єктно-орієнтоване програмування».

Наведено хід виконання роботи та необхідні теоретичні відомості. Для самостійної роботи студентів в кожній роботі передбачено індивідуальні завдання однакового ступеня складності для групи з 25 студентів. Для самоперевірки пропонуються контрольні питання та список рекомендованої літератури.

Для студентів спеціальності 171 «Електроніка», спеціалізації «Електронні системи» всіх форм навчання.

**ЗМІСТ**

# ВСТУП

Програмне забезпечення постійно вдосконалюється. На зміну процедурним мовам програмування прийшли об’єктно-орієнтовані. Сучасні об’єктно-орієнтовані середовища програмування відрізняються високою функціональністю, а візуальні засоби розробки інтерфейсу користувача значно підвищують швидкість та ефективність процесу розробки програмного забезпечення.

Дані методичні вказівки складені у відповідності до програми дисципліни «Об’єктно-орієнтоване програмування». Розглянуто 7 практикумів, присвячених ключовим аспектам розробки програмного забезпечення у середивищі Borland C++ Builder з використанням можливостей графічного користувацького інтерфейсу Windows. Для кожної роботи наводиться хід її виконання та необхідні теоретичні відомості. Всі роз’яснення підкріплюються рисунками.

Для самостійної роботи в кожній роботі передбачено індивідуальні завдання для групи з 25 студентів.

Для самоперевірки пропонуються контрольні питання та список рекомендованої літератури

Основна мета посібника – надати студентові допомогу в опануванні теоретичних положень курсу та в оволодінні основними принципами та прийомами програмування у середовищі швидкої розробки додатків C++Builder, що вивчаються навчальною дисципліною.

Для студентів спеціальності 171 «Електроніка», спеціалізації «Електронні системи» всіх форм навчання.

## КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 1

**Тема: Розробка та виконання додатків**

**Мета:** знайомство з технологією розробки та виконання додатків у середовищі C++ Builder.

### I. Хід виконання роботи.

Створіть на диску **E:** новий проект та збережіть його. Для цього оберіть пункт меню **File/Save All**.

Спочатку C++Builder запропонує зберегти вихідний файл модуля проекту. Надайте модулю ім`я **WorkMain** (за замовчуванням **Unit1.cpp)**.

Потім C++Builder запросить ім'я файлу проекту. Надайте проекту ім'я **Work** (за замовчуванням **Project1.bpr**)..

За замовчанням C++Builder у якості назви вікна форми проекту **Work** використовує ім'я **Form1**. Діючий за замовчанням заголовок можна змінити за допомогою властивості **Caption** форми.

Для того щоб відобразити властивості форми проекту активуйте вікно форми та клацніть на укладці **Properties** вікна **Object Inspector** (рис. 1.1):

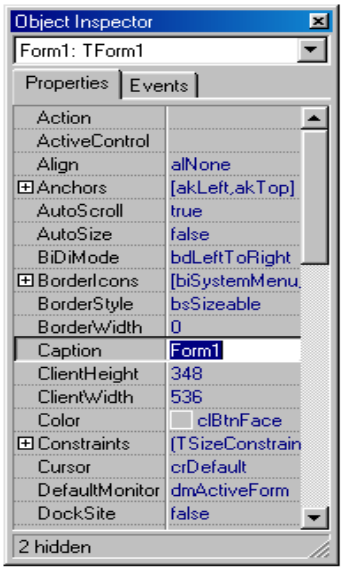


Рис. 1.1. Вікно Object Inspector

☝ Для модернізації будь-якої властивості достатньо лише вивести або обрати нове значення в колонці праворуч від імені властивості.

Замініть, наприклад, значення **Caption** з **Form1** на **Working with Application**.

Збережіть проект. Ви щойно закінчили програмування свого першого додатку.

Запустіть програму на виконання.

Перейдемо до розробки додатка з візуальними компонентами.

Помістіть на форму компонент **Button (**укладка **Standard)**. Переконайтесь, що кнопка активна - вона повинна бути обмежена квадратними маркерами. За замовченням кнопка буде мати ім`я **Button1.** Для того, щоб змінити це ім`я, оберіть властивість **Name** у вікні **Object Inspector** та задайте ім`я **CloseButton**.

- Ім`я компонента повинно складатися з одного слова (без пробілів).

Зверніть увагу, що значення властивості **Caption** і текст кнопки також змінюються на значення **CloseButton**. Для того, щоб властивості **Name** і **Caption** мали різні значення, активуйте властивість **Caption** та встановіть її в значення **Close**. Тепер кнопка буде мати вигляд як на рис. 1.2:

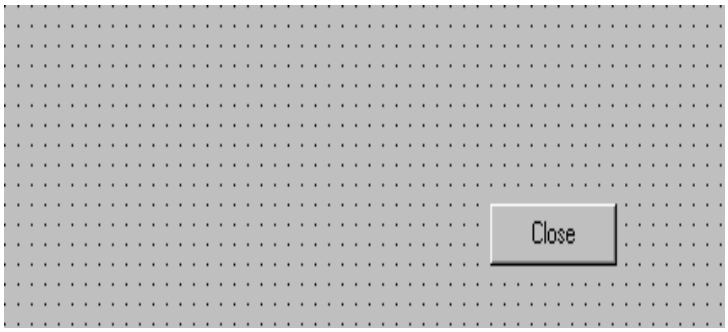


Рис. 1.2. Вигляд кнопки у редакторі форми

Активуйте кнопку **CloseButton** та створить для неї оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TForm1::CloseButtonClick(TObject *Sender)
{
Form1->Close();
}
//---------------------------------------------------------------------------
```

- Тут і в подальшому код, який необхідно додати, виділено **жирним шрифтом**, а решту коду згенеровано Builder-ом.

Збережіть проект. Запустіть програму на виконання.

Модифікуємо проект.

Активуйте кнопку **CloseButton**. Встановіть її властивість **Default** у значення **true.** Таким чином ви обираєте керуючий елемент вікна, який буде активним за замовченням при запуску додатку на виконання.

Запустіть програму на виконання. Для завершення програми натисніть **<Enter>**.

Активуйте кнопку **CloseButton**. Встановіть її властивість **Cancel** у значення **true** (визначає можливість завершення програми натисканням клавіші **<Esc>)**.

Запустіть програму на виконання. Для завершення програми натисніть **<Esc>**.

☝ Запам`ятайте, що лише один компонент форми може мати властивості **Default** або **Cancel** із значенням **true**.

Модифікуємо проект так, щоб при виході з програми відбувався діалог з користувачем.

При виході з додатку викликається метод **Close.** Якщо зачинення вікна можливе, метод **Close** викликає функцію **CloseQuery**, яка повертає значення **true**. Якщо ж функція **CloseQuery** повертає значення **false**, виклик методу **Close** анулюється та вікно залишається відчиненим. Якщо **CloseQuery** повертає **true**, програма викликає обробник подій **OnClose** даної форми.

Активуйте форму **TForm1** та створіть для неї оброблювач події **OnCloseQuery**:

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCloseQuery(TObject *Sender, bool &CanClose)
{
 if(MessageDlg("Terminate the program?", mtConfirmation,
            TMsgDlgButtons() << mbYes << mbNo,0)==mrYes)
   CanClose=true;
 else
   CanClose=false;
}
//---------------------------------------------------------------------------
```

Збережіть проект. Запустіть програму на виконання.

Модифікуємо проект.

Помістіть на форму компонент **Label** (укладка **Standard**) та ще одну кнопку **Button** (укладка **Standard**).

Встановіть властивості компонентів відповідно до таблиці:

| **Компонент** | **Name** | **Caption** |
|---|---|---|
| Label1 | TextLabel | Text |
| Button1 | TextButton | Set |

Форма матиме вигляд як на рис. 1.3:

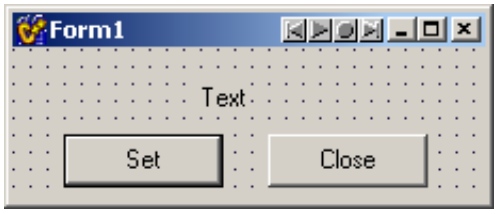


Рис. 1.3. Вікно програмного додатку

Активуйте кнопку **TextButton** та створіть для неї обробник події **OnClick**:

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 Form1->TextLabel->Caption="New text";
}
//---------------------------------------------------------------------------
```

Збережіть проект.

Запустіть програму на виконання..

☝ Після закінчення роботи перепишіть необхідні файли на особистий носій інформації.

**II. Контрольні завдання**

1. Помістіть на форму кнопку, підключіть її обробник подій так, щоб при натисканні на кнопку мінялася назва кнопки.
2. Помістіть на форму кнопку, підключіть її обробник подій так, щоб при натисканні на кнопку з'являлось діалогове вікно з запитом про закриття вікна.
3. Змініть програму таким чином, щоб при натисненні на **TextButton** змінювалась назва вікна.
4. Помістіть на форму позначку, підключіть її обробник подій так, щоб при натисканні на позначку мінялася назва позначки.
5. Помістіть на форму позначку, підключіть її обробник подій так, щоб при натисканні на неї з'являлось діалогове вікно з запитом про закриття вікна.
6. Помістіть на форму ще дві кнопки, підключіть обробник події однієї з кнопок так, щоб при натисканні назва цієї кнопки замінювалося назвою другої кнопки.
7. Помістіть на форму кнопку, підключіть її обробник подій так, щоб при натисканні на кнопку форма зміщувалася на 5 позицій праворуч і на 10 позицій вгору.
8. Помістіть на форму кнопку, підключіть її обробник подій так, щоб при натисканні на кнопку колір всіх написів на формі мінявся на червоний.
9. Підключіть обробник події так, щоб при натисканні на форму вона згорталася на панель задач.
10. Помістіть на форму позначку, підключіть її обробник подій так, щоб при натисканні на неї колір вікна мінявся на чорний, а колір тексту всіх позначок – на білий .

11. Підключіть обробник події так, щоб при натисканні на форму її розміри зменшувались у двічі.
12. Помістіть на форму ще дві кнопки, підключіть обробник події однієї з кнопок так, щоб при натисканні друга кнопка зникала.
13. Модифікуйте програму так, щоб розмір шрифту всіх позначок форми автоматично збільшувався з 8 до 14 пт кожні 3 секунди.
14. Помістіть на форму кнопку, підключіть її обробник подій так, щоб при натисканні на кнопку її ширина збільшувалась, а висота зменшувалась у два рази.
15. Підключіть обробник події так, щоб при деактивуванні форми її заголовок змінювався на «Не активна».
16. Помістіть на форму ще дві кнопки, підключіть обробник події однієї з кнопок, так щоб при натисканні обидві кнопки зникала.
17. Помістіть на форму позначку, підключіть її обробник подій так, щоб вона відображувала час роботи програми.
18. Помістіть на форму кнопку, підключіть її обробник подій , щоб при натисканні на кнопку вона зникала.
19. Помістіть на форму позначку, підключіть її обробник подій так, щоб при натисканні на позначку вона зникала.
20. Помістіть на форму кнопку, підключіть її обробник подій так, щоб при натисканні на кнопку форма зникала.
21. Помістіть на форму позначку, підключіть її обробник подій так, щоб при натисканні на позначку форма зникала.
22. Помістіть на форму позначку, підключіть її обробник подій так, щоб при натисканні на неї мінялася назва **TextButton**.
23. Модифікуйте програму таким чином, щоб час на позначці Label відображався автоматично.
24. Помістіть на форму кнопку для регулювання розміру шрифту позначки.
25. Помістіть на форму позначку, підключіть її обробник подій так, щоб при натисканні на неї форма здвигалася на 15 позицій ліворуч і на 10 позицій вниз.

**III. Контрольні питання.**

1. Що визначає властивість **Name** компонента?
2. Що визначає властивість **Caption** компонента?
3. Які файли треба зберігати при перенесенні додатка на інший комп`ютер?
4. Назвіть основні властивості компонента **Button**.

5. Назвіть основні властивості компонента **Label**.
6. Назвіть основні події компонента **Button**.
7. Назвіть основні події компонента **Label**.
8. Поясніть призначення функції **MessageDlg().**
9. Що визначає функція **Now()**?
10. Назвіть функції, що перетворюють змінну типу **TDateTime** в рядок.

**IV. Рекомендована література:**


1. Архангельский А.Я. Программирование в C++Builder / А.Я. Архангельский. – М.: «Издательство БИНОМ», 2003. – С.157-165;
2. Рейсдорф Кент Borland C++Builder. Освой самостоятельно / Кент Рейсдорф, Кен Хендерсон – М.: «Издательство БИНОМ», 1998. – С. 23-26, 167-209, 209-230.
3. Сван Т. Основи программирования в Delphi для Windows 95 / Т. Сван. – К. : «Диалектика», 1996. – С. 17-70.


## КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 2

**Тема: Програмування клавіатури**
**Мета**: розробка додатків з реалізацією реакцій на події клавіатури

**І. Хід виконання роботи.**

**Проект 2.1.**

Створіть на диску **E:** у своєму каталозі новий проект та збережіть його: модулю **Unit1.cpp** надайте ім'я **MainKey.cpp**, а файлу проекту **Project1.bpr** – ім'я **SimpleKey.bpr**.

Помістіть на форму компонент **BitBtn (**укладка **Additional**), сім компонентів **Bevel** (укладка **Additional**) та сім позначок **Label** (укладка **Standard**). Розташування компонентів показано на рис. 2.1:

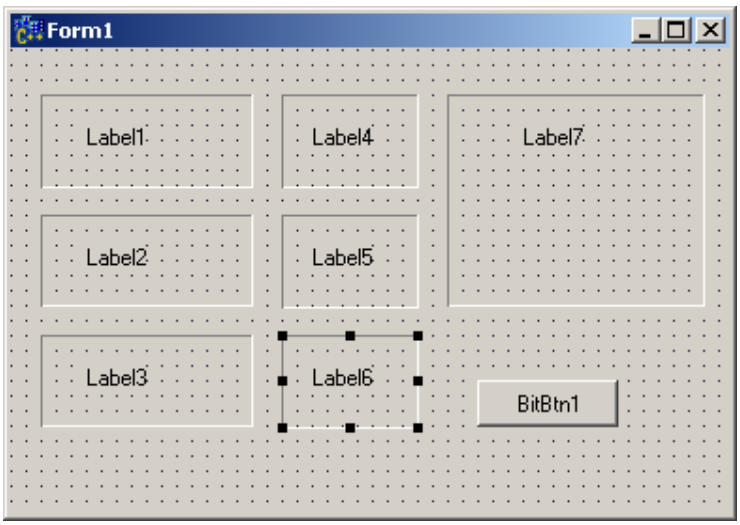


Рис. 2.1. Розташування компонентів

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form1 | MainForm | Caption | Key |
| | | KeyPreview | True |
| Label1 | Label1 | Caption | Keypress character |
| Label2 | Label2 | Caption | Key down value |
| Label3 | Label3 | Caption | Key down shift |
| Label4 | CharLabel | Caption | |
| | | Alignment | taCenter |
| | | AutoSize | false |
| | | Font->Name | Arial |
| | | Font->Size | 20 |
| | | Font->Style | [fsBold]=true |
| Label5 | ValueLabel | Caption | |
| | | Alignment | taCentre |
| | | AutoSize | false |
| | | Font->Name | Arial |
| | | Font->Size | 20 |
| | | Font->Style | [fsBold]=true |
| Label6 | ShiftLabel | Caption | |
| | | Alignment | taCentre |
| | | AutoSize | false |
| | | Font->Name | Arial |
| | | Font->Size | 20 |
| | | Font->Style | [fsBold]=true |
| BitBtn1 | CloseBitBtn | Kind | bkClose |

| | | | |
|---|---|---|---|
| | | TabStop | false |
| Label7 | Label7 | Caption | Press any key and hold down SHIFT, ALT and CTRL keys to see their ASCII and virtual key values |
| | | AutoSize | false |
| | | Font->Name | Arial |
| | | Font->Size | 20 |
| | | Font->Style | [fsBold]=true |
| | | WordWrap | true |

У файл модуля **MainKey.cpp** пропишіть оголошення масиву, який буде використаний у вашій програмі:

```
//---------------------------------------------------------------------------
// масив призначений для відображення на екрані функціональних клавіш
AnsiString FunctionKeys[10]={"F1","F2","F3","F4","F5","F6",
"F7","F8","F9","F10"};
//---------------------------------------------------------------------------
```

Активуйте форму **MainForm** і створіть для неї оброблювачі подій **OnKeyPress**, **OnKeyDown** та **OnKeyUp:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormKeyDown(TObject *Sender, WORD &Key, TShiftState Shift)
{
AnsiString s;
MainForm->ValueLabel->Caption=IntToStr(Key);
s=""; // ініціалізація змінної s порожнім рядком для подальших операторів if
 if(Shift.Contains(ssShift))
    s+="Shift+";            //Shift
if(Shift.Contains(ssAlt))
    s+="Alt+";              //Alt
if(Shift.Contains(ssCtrl))
    s+="Ctrl+";             //Ctrl
 if(s.Length())
  {
    s.Delete(s.Length(),1);
   MainForm->ShiftLabel->Caption=s;
```

```
  }
 if(Key>=VK_F1 && Key<=VK_F10)
   MainForm->CharLabel->Caption=FunctionKeys[Key-VK_F1];
else
   MainForm->CharLabel->Caption="";   //очищення старого значення
if(Key==VK_SPACE)
   Key=0;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormKeyUp(TObject *Sender, WORD &Key,
 TShiftState Shift)
{
 MainForm->CharLabel->Caption="";
 MainForm->ValueLabel->Caption="";
 MainForm->ShiftLabel->Caption="";
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormKeyPress(TObject *Sender, char &Key)
{
if (Key==8)
MainForm->CharLabel->Caption=”BackSpace”;
if (Key==9)
MainForm->CharLabel->Caption=”Tab”;
if (Key==13)
MainForm->CharLabel->Caption=”Enter”;
if (Key==27)
MainForm->CharLabel->Caption=”Esc”;
MainForm->CharLabel->Caption= Key;
MainForm->ValueLabel->Caption= IntToStr(Key);
}
```

Збережіть проект. Запустіть програму на виконання (рис. 2.2):

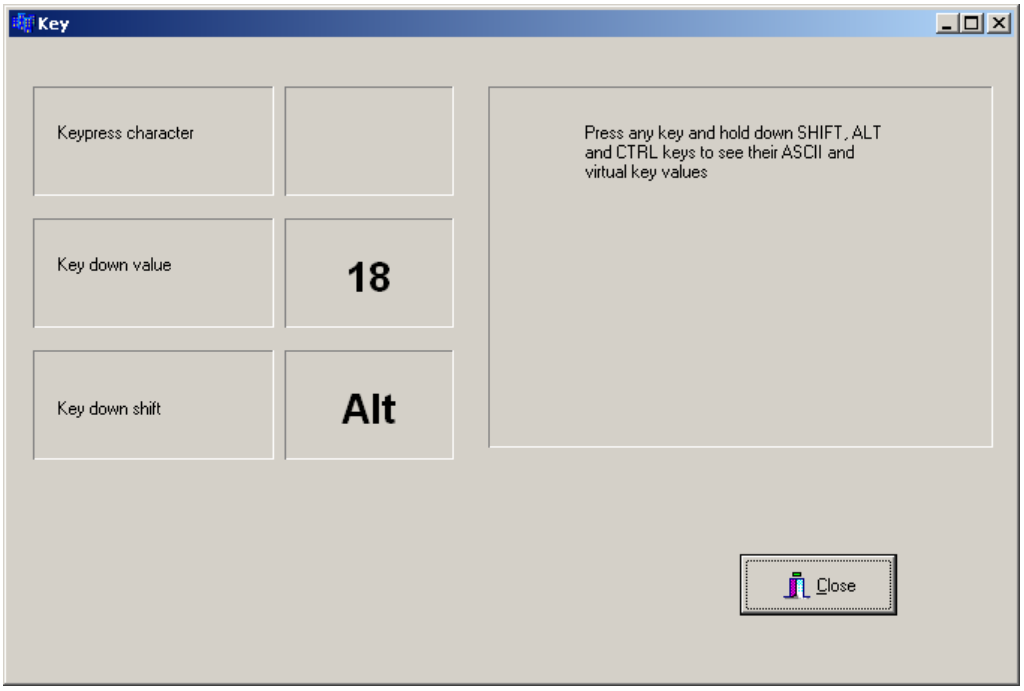


Рис. 2.2. Вікно програмного доданку проекту 2.1

Натискаючи різноманітні клавіші, у тому числі й функціональні, звертайте увагу на значення, що виводяться у позначках.

**Key press character** відображає символи ASCII. Для клавіш, що не мають видимого символу, це вікно залишається порожнім.

**Key down value** виводить числовий код клавіші. При натисканні комбінацій клавіш у цьому вікні з'явиться значення клавіші, яка була натиснута останньою.

**Key down shift** показує стан спеціальних клавіш.

Запишіть результати виконання програми до таблиці:

| **Клавіша** | **Код** | **Клавіша** | **Код** |
|---|---|---|---|
| Enter | | BackSpace | |
| 1 | | Del | |
| ! | | Ins | |
| Esc | | + | |
| / | | R | |
| < | | . | |
| } | | m | |
| 7 | | Э | |
| ( | | Z | |
| 9 | | Shift+Alt | |

**Проект 2.2.**

Створіть на диску **E:**\ у своєму каталозі новий проект та збережіть його: модулю **(Unit1.cpp)** надайте і'мя **IndMain.cpp**, а файлу проекту

(**Project.bpr**) - ім'я **Indicate.bpr**.

Проект передбачає створення додатку, який містить рядок стану з індикаторами клавіш **CapsLock, NumLock, ScrollLock** та **Ins** клавіатури.

Помістіть на форму компонент **Bevel** (укладка **Additional),** на ньому встановіть позначку **Label** (укладка **Standard)**. У нижній частині форми розмістіть компонент **Panel (**укладка **Standard),** а на ньому ще чотири компоненти **Panel.**

Розташування компонентів показано на рис. 2.3:

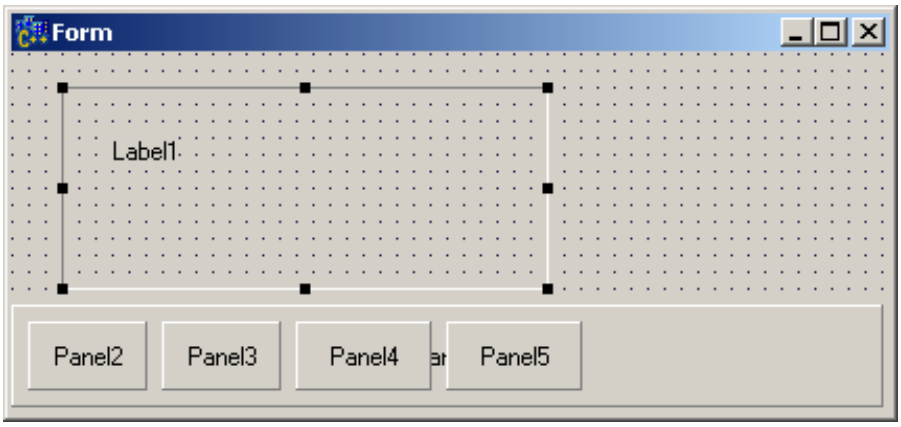


Рис. 2.3. Розташування компонентів проекту 2.2

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form1 | MainForm | Caption | Key Indicator |
| | | KeyPreview | True |
| Panel1 | StatusPanel | Caption | |
| | | BevelInner | bvLowered |
| | | BevelOuter | bvRaised |
| | | Height | 30 |
| | | Align | alBottom |
| Label1 | Label1 | Alignment | taCenter |
| | | AutoSize | false |
| | | WordWrap | true |
| | | Caption | Press the CapsLock, NumLock, ScrollLock and Ins keys |
| | | Font.Name | Arial |
| | | Font.Size | 20 |
| | | Font.Style | [fsBold]=true |

| | | | |
|---|---|---|---|
| Panel2 | CapsLockPanel | Caption | |
| | | Align | alLeft |
| | | BevelInner | bvLowered |
| | | BevelOuter | bvNone |
| Panel3 | NumLockPanel | Caption | |
| | | Align | alLeft |
| | | BevelInner | bvLowered |
| | | BevelOuter | bvNone |
| Panel4 | ScrollLockPanel | Caption | |
| | | Align | alLeft |
| | | BevelInner | bvLowered |
| | | BevelOuter | bvNone |
| Panel5 | InsPanel | Caption | |
| | | Align | alLeft |
| | | BevelInner | bvLowered |
| | | BevelOuter | bvNone |

Модифікуємо проект так, щоб індикатори клавіш у стані „вимкнено” відображались текстом із зниженою яскравістю, а у стані „включено” - в нормальному виді. Компонент **Panel** не може знижувати яскравості тексту своєї властивості **Caption**. Тому в середині кожної панелі помістіть по одній позначці **Label** (укладка **Standard**).

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Caption** | **Enabled** |
|---|---|---|---|
| Label2 | CapsLockLabel | Caps | false |
| Label3 | NumLockLabel | Num | false |
| Label4 | ScrollLockLabel | Scroll | false |
| Label5 | InsLabel | Ins | false |

Помістіть на форму компонент **BitBtn** (стор. **Additional**). Встановіть властивості компонента згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| BitBtn1 | CloseBitBtn | Kind | bkClose |

Форма матиме вигляд як на рис. 2.4:

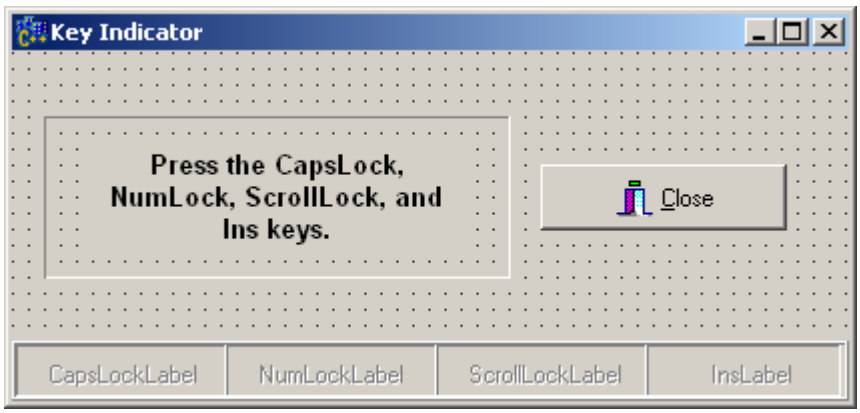

Рис. 2.4. Налаштування властивостей компонентів проекту 2.2

У файл модуля **IndcMain.cpp** пропишіть функцію, яка буде використовуватись у вашій програмі:

```
//---------------------------------------------------------------------------
void UpdateKeyPanel(void)
{
MainForm ->CapsLockLabel->Enabled=GetKeyState(VK_CAPITAL) ;
MainForm->NumLockLabel->Enabled=GetKeyState(VK_NUMLOCK);
MainForm ->ScrollLockLabel->Enabled=GetKeyState(VK_SCROLL) ;
MainForm -> InsLabel->Enabled=GetKeyState(VK_INSERT) ;
}
//---------------------------------------------------------------------------
```

Активуйте форму **MainForm** та створіть для неї оброблювачі подій **OnActivate, OnKeyDown**, **OnKeyUp**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormActivate(TObject *Sender)
{
 UpdateKeyPanel();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormKeyDown(TObject *Sender, WORD &Key,  TShiftState Shift)
{
 UpdateKeyPanel();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormKeyUp(TObject *Sender, WORD &Key,
TShiftState Shift)
{
 UpdateKeyPanel();
}
```

//---------------------------------------------------------------------------

Збережіть проект.

Запустіть програму на виконання (рис. 2.5):

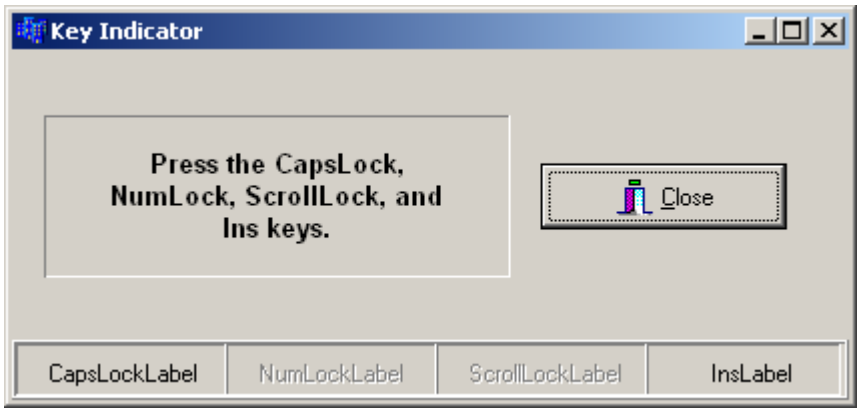


Рис. 2.5. Вікно програмного додатку проекту 2.2

**Проект 2.3**.

Створіть на диску **E:** у своєму каталозі новий проект: модулю **Unit1.cpp** надайте ім'я **EditMain.cpp**, а файлу проекту **Project1.bpr** – ім'я **EditChar.bpr**.

Помістіть на форму два компоненти **Edit** (стор. **Standard**), один компонент **Bevel** (стор. **Additional**), два компоненти **BitBtn** (стор. **Additional**) та чотири компоненти **Label** (укладка **Standard)**:

Розташування компонентів показано на рис. 2.6:

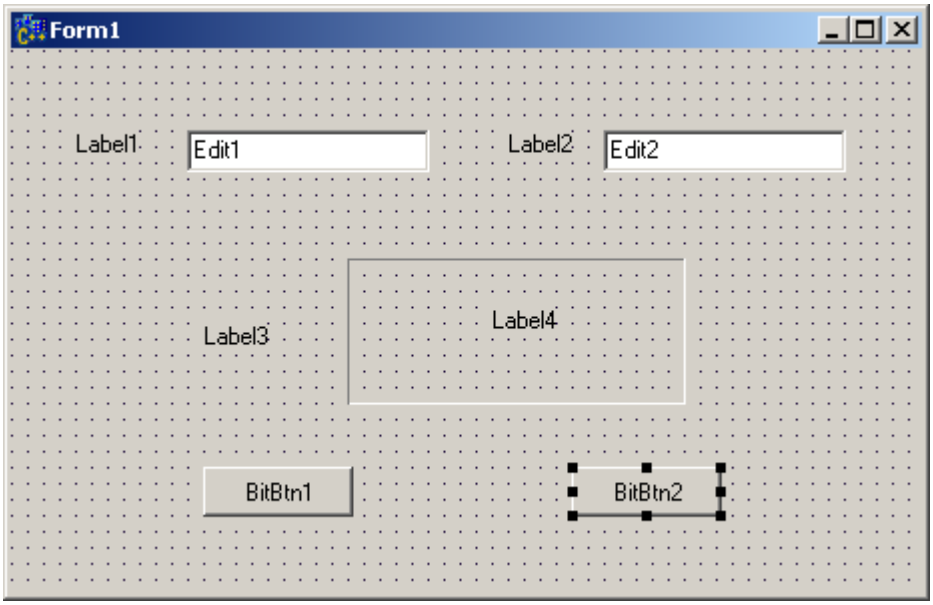


Рис. 2.6. Розташування компонентів проекту 2.3

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Ім`я(Name)** | **Властивість** | **Значення** |
|---|---|---|---|
| Form1 | MainForm | Caption | Edit Char |
| Bevel1 | Bevel1 | | |

| | | | |
|---|---|---|---|
| Label1 | Item1 | Caption | Item1 |
| Label2 | Item2 | Caption | Item2 |
| Label3 | SumLabel | Caption | Sum |
| Label4 | SumResultLabel | Caption | |
| Edit1 | Item1Edit | Text | |
| Edit2 | Item2Edit | Text | |
| BitBtn1 | SumBitBtn | Caption | Sum |
| BitBtn2 | CloseBitBtn | Kind | bkClose |

Активуйте компонент **SumBitBtn** і створіть для нього оброблювач події **OnClick**.

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::SumBitBtnClick(TObject *Sender)
{
 float Item1,Item2;

 Item1=StrToFloat(MainForm->Item1Edit->Text);
 Item2=StrToFloat(MainForm->Item2Edit->Text);
 MainForm->SumResultLabel-
>Caption=FormatFloat("0.00",Item1+Item2);
}
//---------------------------------------------------------------------------
```

Функція **FormatFloat** здійснює форматований вивід чисел з плаваючою комою (перетворює числа типу **float** у **AnsiString**).

Рядок **“0.00”** вказує формат виведення чисел. Існують наступні специфікатори формату:

**0** – якщо величина має в цій позиції цифру, то встановлюється ця цифра, інакше – встановлюється 0.

**#** – якщо величина має в цій позиції цифру, то встановлюється ця цифра, інакше –не встановлюється нічого.

**.** – поле для десяткової крапки. Сюди встановлюється символ, що заданий константою **DecimalSeparator**.

**,** – поле для роздільника тисяч.

**E+, E-, e+, e-** – ознака того, що число буде перетворено з характеристикою та мантисою.

**;** – розділяють специфікатори формату для додатних, від’ємних і нульових чисел.

У поле компонента **Edit** можна вводити лише символи. Перетворення символів у число типу **float** здійснюється за допомогою функції **StrToFloat()**.

Встановіть специфікатор формату **“0.00e-0”**. Виконайте обчислення та запишіть результат до таблиці в колонці **Sum1**.

Встановіть специфікатор формату **“#.##e-0”**. Виконайте обчислення та запишіть результати до таблиці в колонці **Sum2**:

| **Item1** | **Item2** | **Sum1** | **Sum2** |
|---|---|---|---|
| 0,1 | 0,2 | | |
| 1,45 | 1,23 | | |
| 1,2e-3 | 2,7e-5 | | |
| 6,1e4 | 9,1e3 | | |
| -1,89e-4 | -8,99e-3 | | |
| 1,89e-4 | 8,99e-3 | | |
| 8,45e6 | 3,12e5 | | |
| 0,002 | 0,0045 | | |

Модифікуємо програму так, щоб можливим було лише введення цифр, десяткової коми та символів **“E”**, **“e”**, **“+”**, **“-“**.

Активуйте компонент **Item1Edit** і створіть для нього оброблювач події **OnKeyPress**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::Item1EditKeyPress(TObject *Sender, char &Key)
{
 if(Key=='e' || Key=='E' || Key=='+' || Key=='-' || Key==',' || (Key>='0' && Key<='9'))
  {
    return;
  }
 else
  {
    Key=0;
  }
}
//---------------------------------------------------------------------------
```

Для компонента **Item2Edit** зробіть у коді програми аналогічні зміни.

Збережіть проект.

Запустіть програму на виконання. Спробуйте ввести заборонені символи.

У Windows використовується поняття *фокус введення* для того, щоб визначити, куди посилати події клавіатури.

Обробник події **OnExit** викликається при втраті компонентом фокусу введення, а обробник події **OnEnter** при одержанні компонентом фокусу введення.

Модифікуємо програму так, щоб компонент редагування змінював свій колір при одержанні ним фокусу введення та відновлював його при втраті цим компонентом фокусу введення.

Активуйте компонент **Item1Edit** і створіть для нього оброблювач події **OnEnter**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::Item1EditEnter(TObject *Sender)
{
 MainForm->Item1Edit->Color=clRed;
}
//---------------------------------------------------------------------------
```

Для компонента **Item2Edit** зробіть у коді програми аналогічні зміни.

Активуйте компонент **Item1Edit** і створіть для нього оброблювач події **OnExit**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::Item1EditExit(TObject *Sender)
{
 MainForm->Item1Edit->Color=clWindow;
 if(MainForm->Item1Edit->Text=="")
  {
    MainForm->Item1Edit->Text="0";
  }
}
//---------------------------------------------------------------------------
```

Для компонента **Item2Edit** зробіть у коді програми аналогічні зміни.

Збережіть проект.

Запустіть програму на виконання (рис. 2.7):

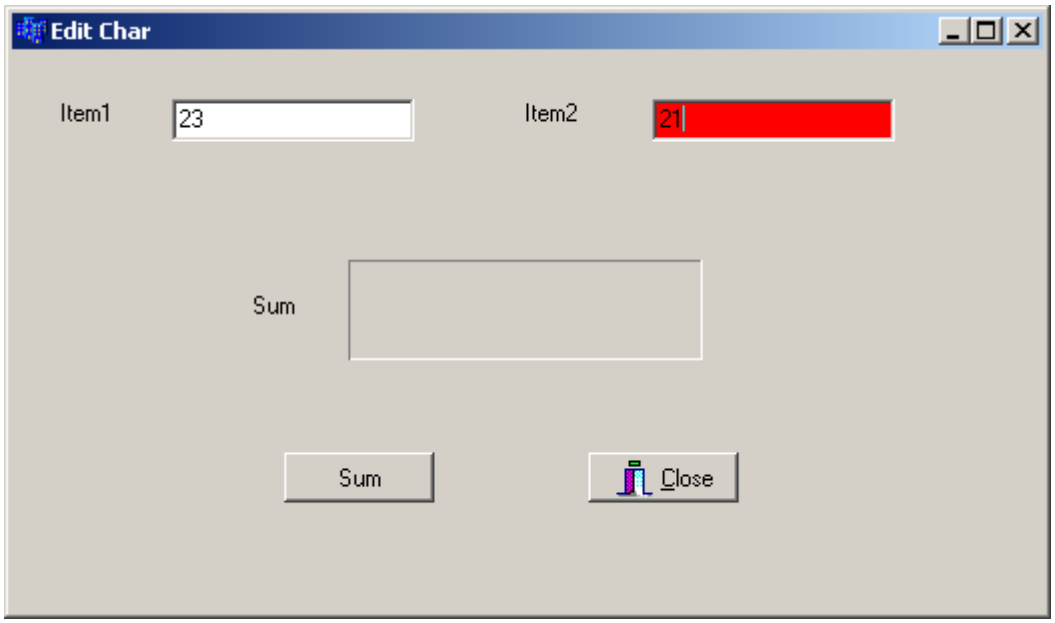

Рис. 2.7. Вікно програмного додатку проекту 2.3

**II. Контрольні завдання.**

1. Змініть обробник події OnKeyPress так, щоб при натисканні комбінації клавіш Ctrl+H висота форми збільшувалась вдвічі.
2. Змініть обробник події OnKeyPress так, щоб при натисканні комбінації клавіш Ctrl+K форма зсувалася ліворуч на 20 пікселів.
3. Змініть обробник події OnKeyDown так, щоб при натисканні комбінації клавіш Ctrl+P форма ставала жовтого кольору.
4. Змініть обробник події OnKeyDown так, щоб при натисканні комбінації клавіш Ctrl+Q додаток закривався.
5. Змініть обробник події OnKeyDown так, щоб при натисканні комбінації клавіш Shift+P форма ставала червоного кольору.
6. Змініть обробник події OnKeyDown так, щоб при натисканні комбінації клавіш Shift+M у формі ставала неактивною кнопка мінімізування форми.
7. Змініть обробник події OnKeyDown так, щоб при натисканні комбінації клавіш Shift+Z розміри форми збільшувались на 50 пікселів.
8. Змініть обробник події OnKeyDown так, щоб при натисканні комбінації клавіш Shift+P форма ставала квадратною за розміром більшої сторони.
9. Змініть обробник події OnKeyDown так, щоб при натисканні клавіші F1 властивість Caption форми приймала значення «Допомога».
10. Змініть обробник події OnKeyDown так, щоб при натисканні клавіші F2 над формою зникав курсор.
11. Змініть обробник події OnKeyDown так, щоб при натисканні клавіші F3 форма перемістилась у лівий верхній кут екрана.
12. Змініть обробник події OnKeyDown так, щоб при натисканні клавіші F5

форма зникала з екрану.

13. Змініть обробник події OnKeyDown так, щоб при натисканні клавіші F6 форма ставала чорного кольору.
14. Змініть обробник події OnKeyDown так, щоб при натисканні клавіші F7 форма ставала неактивною.
15. Модифікуйте програму так, щоб при активному Caps Lock форма ставала червоного кольору, а в іншому випадку – білого.
16. Модифікуйте програму так, щоб при активному Caps Lock форма ставала квадратною (за більшою стороною), а в іншому випадку – поверталась до попереднього розміру.
17. Модифікуйте програму так, щоб при активному Num Lock форма ставала квадратною (зі стороною, що дорівнює середньому арифметичному між висотою та шириною форми), а в іншому випадку поверталась до попереднього розміру.
18. Модифікуйте програму так, щоб при активному Scroll Lock форма перемішалася в лівий нижній кут екрана, а в іншому випадку – у правий верхній.
19. Модифікуйте програму так, щоб при активних Caps Lock + Num Lock форма ставала червоною і перемішувалась у лівий верхній кут екрану.
20. Модифікуйте програму так, щоб при активних Insert + Num Lock форма ставала зеленою і квадратною.
21. Модифікуйте програму так, щоб при активних Scroll Lock + Scroll Lock форма ставала жовтою і змінювався заголовок форми.
22. Модифікуйте програму так, щоб при натисненні клавіші «G» та активному Caps Lock додаток закривався.
23. Модифікуйте програму так, щоб при натисненні клавіші «G» та активному Num Lock форма ставала неактивною.
24. Модифікуйте програму так, щоб при натисненні клавіші «G» та активному Insert форма ставала неактивною.
25. Модифікуйте програму так, щоб при активному Scroll Lock форма закривалася.

**III. Контрольні питання.**

1. Яка властивість форми забезпечує реакцію на події клавіатури?
2. Коли виникає подія OnKeyPress компонента?
3. Коли виникає подія **OnKeyDown** компонента**?**
4. Коли виникає подія **OnKeyUp** компонента**?**
5. Коли виникає подія **OnEnter** компонента?
6. Коли виникає подія **OnExit** компонента?
7. Перерахуйте специфікації формату функції **FormatFloat()**.

8. Як заборонити опрацювання деякої клавіші?
9. Яка функція перетворює ціле число у рядок?
10. Яка функція перетворює рядок у число типу float?
11. Назвіть призначення компонента BitBtn.
12. Перерахуйте основні властивості компонента BitBtn.

**IV. Рекомендована література:**


1. Архангельский А.Я. Программирование в C++Builder / А.Я. Архангельский – М.: «Издательство БИНОМ», 2003. – С.157-165.
2. Архангельский А.Я. C++Builder 6. Справочное пособие. Книга 2. Классы и компоненты / А.Я. Архангельский – М.: Бином-Пресс, 2002. – 528 с.: ил.
3. Рейсдорф Кент Borland C++Builder. Освой самостоятельно / Кент Рейсдорф, Кен Хендерсон – М.: «Издательство БИНОМ», 1998. –С .230-247.
4. Сван Т. Основи программирования в Delphi для Windows 95 / Т. Сван. – К. : «Диалектика», 1996. –480 с., ил. – С. 71-94.


## КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 3

**Тема: Створення списків**

**Мета:** розробка додатків з простими та багатосторінковими списками з реалізацією обробки текстових файлів.

**І. Хід виконання роботи.**

**Проект 3.1.**

Створіть на диску **E:**\ у своєму каталозі новий проект та збережіть його**: м**одулю **Unit1.cpp** надайте им'я **StrMain.cpp**, а файлу проекту **Project1.bpr** – им'я **SetStr.bpr**.

Розташуйте на формі один компонент **ListBox** (укладка **Standard**), один компонент **Edit** (укладка **Standard**), дві позначки **Label** (укладка **Standard**), сім кнопок **Button** (укладка **Standard**) та одну кнопку **BitBtn (**укладка **Additional).** Розташування компонентів показано на рис. 1:

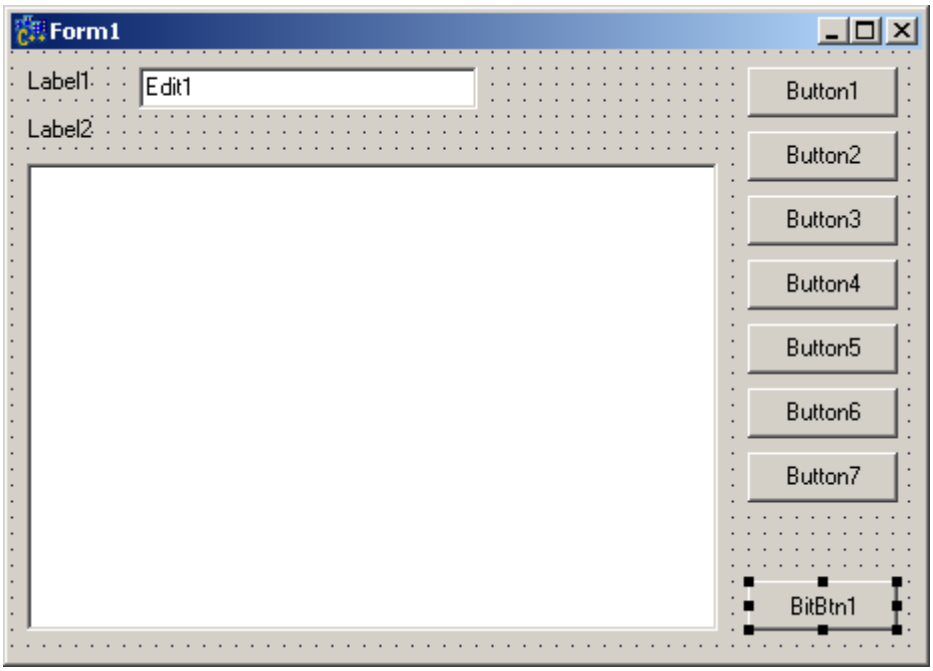


Рис. 3.1. Розташування компонентів поекту 3.1

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form1 | MainForm | Caption | String |
| ListBox1 | StrListBox | | |
| Label1 | Label1 | Caption | String: |
| Label2 | Label2 | Caption | List Box: |
| Button1 | SetButton | Caption | Set |
| Button2 | InsertButton | Caption | Insert |
| Button3 | DeleteButton | Caption | Delete |
| Button4 | ClearButton | Caption | Clear |
| Button5 | SaveButton | Caption | Save |
| Button6 | ReadButton | Caption | Read |
| Button7 | RestoreButton | Caption | Restore |
| Edit1 | StrEdit | Text | |
| BitBtn1 | CloseBitBtn | Kind | bkClose |

Дані до списку компонента **ListBox** (як об’єкта класу **TStringList**) можна вносити як під час розробки додатку, так й під час його виконання.

В процесі розробки додатку клікніть на кнопці підстановки властивості **Items**, щоб відкрити вікно Редактора **String List Editor** та введіть елементи вашого списку з клавіатури або з файлу.

Для додавання рядків до списку під час роботи програми треба викликати відповідні методи властивості **Items**.

У файл модуля **StrMain.cpp** пропишіть змінну, яка буде використана у вашій програмі:

```
//---------------------------------------------------------------------------
TStringList *StringL;
//---------------------------------------------------------------------------
```

У заголовному файлі **StrMain.h** після директиви private пропишіть змінну, яка буде використана у вашій програмі:

```
//---------------------------------------------------------------------------
private:        // User declarations
        AnsiString CurDir;
//---------------------------------------------------------------------------
```

Активуйте форму **MainForm** та створіть для неї оброблювачі подій **OnCreate та OnDestroy:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
StringL=new  TStringList;
CurDir=ExtractFilePath(Application->ExeName);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormDestroy(TObject *Sender)
{
delete StringL;
}
//---------------------------------------------------------------------------
```

Для кнопок **Button** створіть оброблювачі події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::SetButtonClick(TObject *Sender)
{
 if (StrEdit->GetTextLen() > 0)
 StrListBox->Items->Add(StrEdit->Text);
 StringL->Add(StrEdit->Text);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::InsertButtonClick(TObject *Sender)
{
 if (StrEdit->GetTextLen() > 0)
 StrListBox->Items->Insert(StrListBox->ItemIndex, StrEdit->Text);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::DeleteButtonClick(TObject *Sender)
```

```
{
 StrListBox->Items->Delete(StrListBox->ItemIndex);
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::ClearButtonClick(TObject *Sender)
{
 StrListBox->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::SaveButtonClick(TObject *Sender)
{
 StrListBox->Items->SaveToFile(CurDir+"String.txt");
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::ReadButtonClick(TObject *Sender)
{
 try
  {
   StrListBox->Items->LoadFromFile(CurDir+"String.txt");
  }
 catch(...)
  {
   MessageDlg("File not exist!", mtError, TMsgDlgButtons() << mbOK, 0);
  }
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::RestoreButtonClick(TObject *Sender)
{
 StrListBox-> Items->Assign(StringL);
}
//---------------------------------------------------------------------------
```

Активуйте форму **MainForm** і створіть для неї оброблювач події **OnCloseQuery**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormCloseQuery(TObject *Sender, bool &CanClose)
{
 if (StrListBox->Items->Count > 0)
  {
```

```
    if (MessageDlg("Save Data?", mtConfirmation, TMsgDlgButtons() <<
                    mbYes << mbNo, 0) != mrNo)
      StrListBox->Items->SaveToFile(CurDir+"String.txt");
  }
}
//---------------------------------------------------------------------------
```

Збережіть проект.

Запустіть програму на виконання (рис. 3.2).

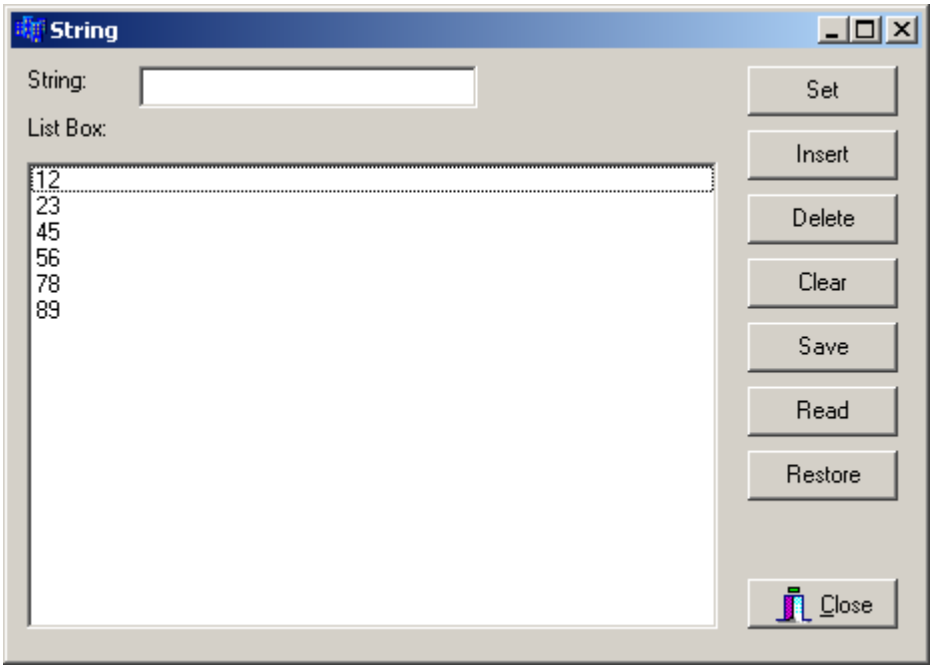


Рис. 3.2. Вікно програмного додатку проекту 3.1

**Проект 3.2.**

Створіть на диску **E:**\ у своєму каталозі новий проект та збережіть його**: м**одулю **Unit1.cpp** надайте им'я **MainBox.cpp**, а файлу проекту **Project1.bpr** – им'я **WinFont.bpr**.

Помістіть на форму один компонент **ListBox** (укладка **Standard**), п'ять позначок **Label** (укладка **Standard**), один компонент **Bevel (**укладка **Additional**), один компонент **SpinEdit (**укладка **Samples**), одну кнопку **BitBtn (**укладка **Additional).**

Розташування компонентів показано на рис. 3.3:

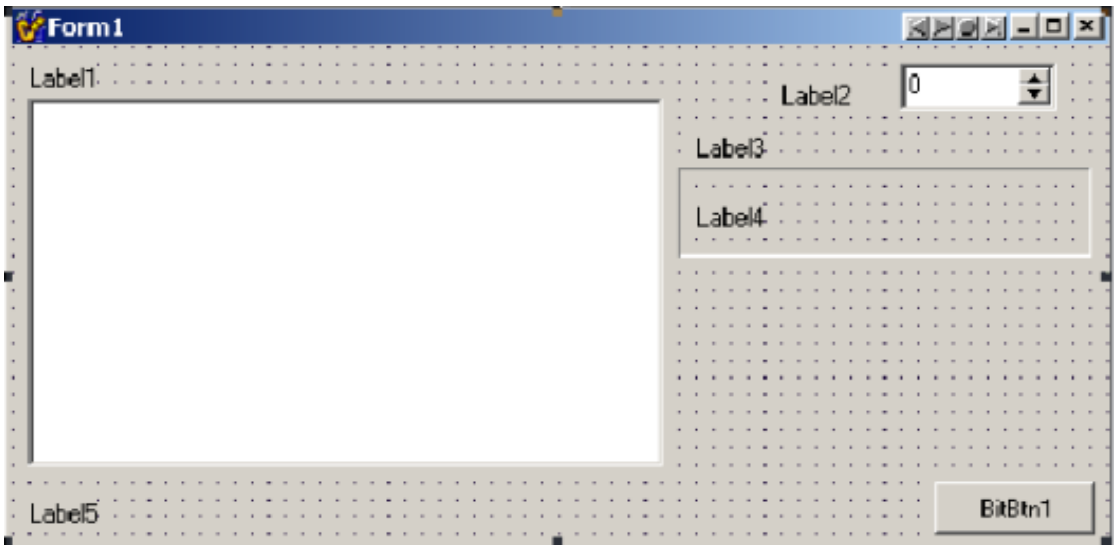


Рис. 3.3. Розташування компонентів проекту 3.2

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form | MainForm | Caption | Fonts List |
| Label | | Caption | Fonts: |
| Label | | Caption | Size: |
| Label | | Caption | Selected font: |
| Label | FontNameLabel | Caption | System |
| Label | SampleLabel | Caption | A B C – 1 2 3 ! |
| ListBox | FontListBox | | |
| SpinEdit | SpinEditSize | MinValue | 8 |
| | | MaxValue | 24 |
| | | Value | 10 |
| BitBtn | CloseBitBtn | Kind | bkClose |

Активуйте форму **MainForm** та створіть оброблювач події **OnCreate** форми. Введіть текст програми відповідно до коду:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
FontListBox->Items=Screen->Fonts;
}
```

C++Builder автоматично створює глобальний об'єкт Screen (типу TScreen), властивості якого визначаються з інформації Windows. Властивість Fonts (типу Strings) об'єкту Screen містить список шрифтів, доступних на даному комп'ютері.

Активуйте компонент **FontListBox** та створіть оброблювачі подій

**OnDblClick** та **OnKeyDown** компонента:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FontListBoxDblClick(TObject *Sender)
{
SampleLabel->Font->Name=FontListBox->Items->
Strings[FontListBox ->ItemIndex];
FontNameLabel->Caption=SampleLabel->Font->Name;
}
//---------------------------------------------------------------------------
```

Модифікуємо проект так, щоб користувачі мали можливість вибирати елементи списку за допомогою клавіш <**Enter**> та <**Space**>. Для цього підключимо ці клавіші до оброблювача події **OnKeyDown** об'єкту **FontListBox:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm:: FontListBoxKeyDown(TObject *Sender,
WORD&Key, TShiftState Shift)
{
if(Key==VK_RETURN || Key==VK_SPACE)
{
TMainForm:: FontListBoxDblClick (Sender);
}
SampleLabel->Font->Size=SpinEditSize->Value;
}
//---------------------------------------------------------------------------
```

Відповідна функція перевіряє належність змінної **Key** множині двох значень **VK_RETURN** та **VK_SPACE.** У разі належності викликається оброблювач **FontListBoxDblClick**. Таким чином при натисканні клавіші введення чи пробілу відбувається імітація події подвійного клацання миші.

Активуйте компонент **SpinEditSize** та створіть оброблювач події **OnChange** компонента:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::SpinEditSizeChange(TObject *Sender)
{
SampleLabel->Font->Size=SpinEditSize->Value;
}
//---------------------------------------------------------------------------
```

Збережіть проект та запустіть його на виконання (рис. 3.3).

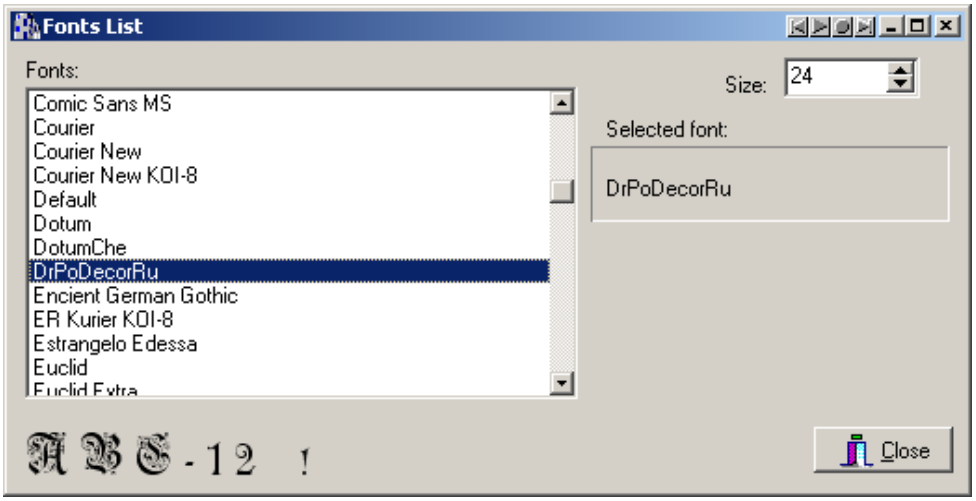


Рис. 3.3. Вікно програмного додатку проекту 3.2

Яке бачимо, програма виводить список усіх наявних у системі шрифтів. При переміщенні списком відображається назва вибраного шрифту та текст, записаний даним шрифтом. Можна змінювати розмір шрифту (вводити з клавіатури або скористатись спеціальним лічильником).

**Проект 3.3.**

Створіть на диску **E:**\ у своєму каталозі новий проект та збережіть його: модулю **Unit1.cpp** надайте ім'я **Main.cpp**, а файлу проекту **Project1.bpr** – им'я **Case1. bpr**.

Новий проект передбачає створення ділового щоденника, що складається з дванадцяти сторінок відповідно до дванадцяти місяців року.

Помістіть на форму компонент **Notebook** (укладка **Win 3.1**), кнопку з растровим зображенням **BitBtn** (укладка **Additional**), компонент **TabSet** (укладка **Win 3.1**) та компонент **GroupBox** (укладка **Standard**). На компоненті **GroupBox** розмістіть один компонент **Edit** (укладка **Standard**), три кнопки **Button** (укладка **Standard**) та одну позначку **Label** (укладка **Standard**).

Розташування компонентів показано на Рис. 3.3:

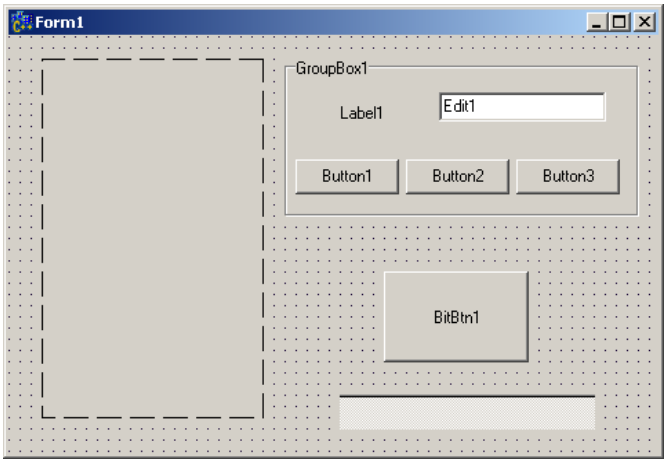


Рис. 3.3. Розташування компонентів проекту 3.3

Встановіть властивості компонентів відповідно до таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form1 | MainForm | Caption | YearBook |
| Edit1 | EditItem | Text | |
| Label1 | Label1 | Caption | Item |
| Button1 | BtnAdd | Caption | &Add |
| Button2 | BtnChange | Caption | &Change |
| Button3 | BtnDelete | Caption | &Delete |
| BitBtn1 | BtnClose | Caption | Close |
| | | Kind | bkClose |
| TabSet1 | TabSet1 | Align | alBottom |
| GroupBox1 | GroupBox1 | Caption | |

Форма матиме вигляд як на рис. 3.4:

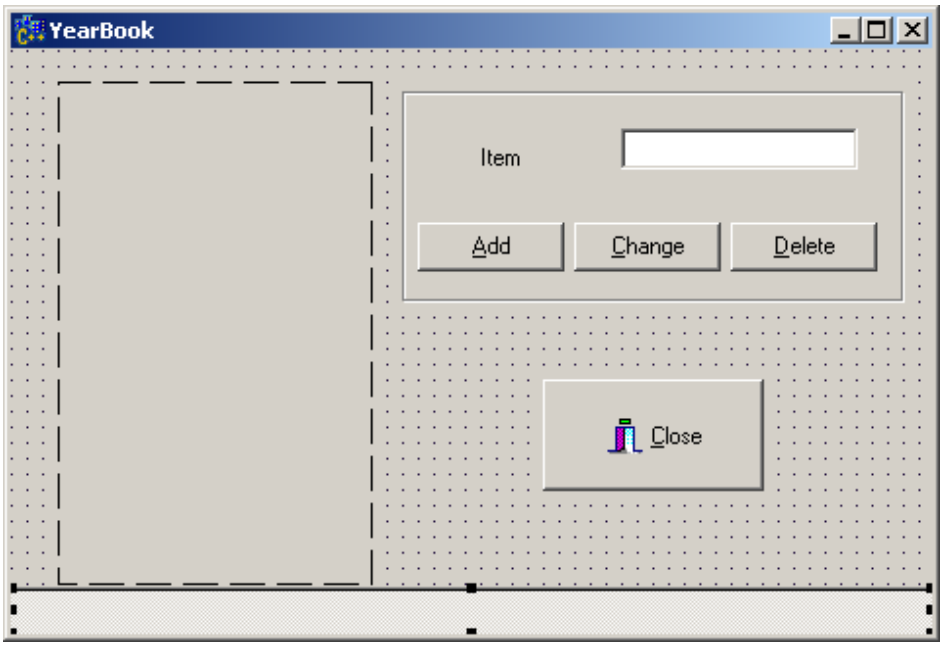


Рис. 3.4. Налаштування властивостей компонентів проекту 3.3

Збережіть проект.

Налаштуйте об’єкт **Notebook**. Подвійним клацанням активуйте його властивість **Pages –** з'явиться вікно Редактора блокнота **Notebook editor**. За замовчуванням у блокноті задана лише одна сторінка під іменем **Default**. Оберіть цю сторінку та клацніть на кнопці **Edit**.

У полі **Page name** діалогового вікна **Edit name,** що з’явилося, напишіть слово **Jan**. Номер контекстної підказки **Help context** залиште зі значенням **0**. Клацніть **ОК** для збереження змін властивостей сторінки.

Клацніть на кнопці **Add.** Знову з’явиться вікно **Edit Page**. Повторіть попередні кроки, створивши сторінки **Feb, Mar**, **Apr**, **May**, **Jun**, **Jul**, **Aug**, **Sep**, **Oct**, **Nov**, **Dec** відповідно.

Закрийте вікно Редактора блокнота **Notebook editor** кнопкою **Close**

Збережіть проект.

На сторінці блокнота **Notebook** розташуйте позначку **Label** (укладка **Standard**) та компонент **ListBox** (укладка **Standard**).

Задайте властивості компонентів відповідно до таблиці:

| **Компонент** | **Ім'я (Name)** | **Властивість** | **Значення** |
|---|---|---|---|
| Notebook1 | NoteBook1 | ActivePage | Jan |
| Label2 | LblJan | Caption | January |
| ListBox1 | LstJan | | |

Форма матиме вигляд як на рис. 3.5:

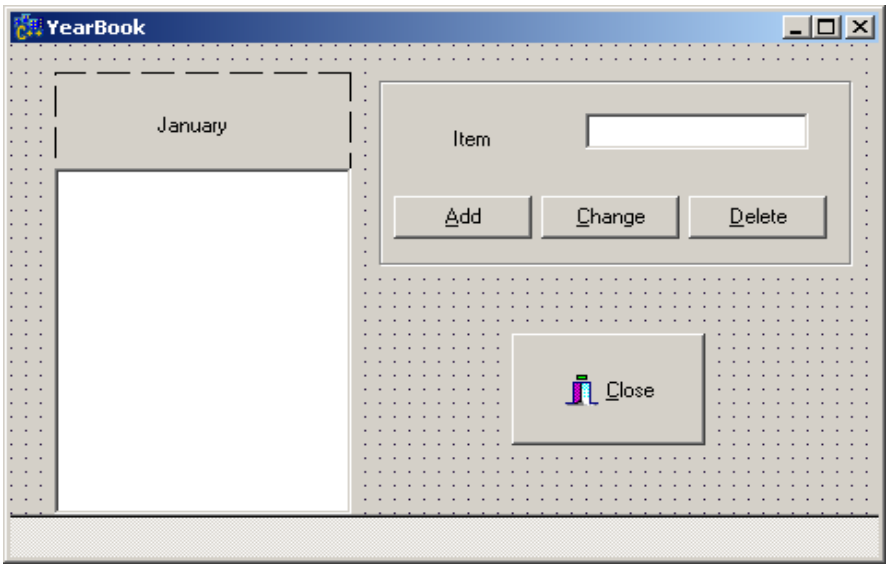


Рис. 3.5. Налаштування об’єкта Notebook

Знову активуйте **Notebook**. Повторіть попередні кроки: на сторінці блокнота розташуйте позначку **Label** та компонент **ListBox**.

Задайте властивості компонентів відповідно до таблиці:

| **Компонент** | **Ім'я (Name)** | **Властивість** | **Значення** |
|---|---|---|---|
| Notebook1 | NoteBook1 | ActivePage | Feb |
| Label2 | LblFeb | Caption | February |
| ListBox1 | LstFeb | | |

Аналогічні дії виконайте для сторінок блокнота **Mar**, **Apr**, **May**, **Jun**, **Jul**, **Aug**, **Sep**, **Oct**, **Nov**, **Dec** відповідно.

Збережіть проект.

Тепер необхідно забезпечити появу закладок компонента **TabSet** під час виконання програми та синхронізацію закладок зі сторінками блокнота **Notebook**.

Активуйте форму **FormMain** та створіть для неї оброблювач події **OnCreate**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
TabSet1->Tabs=NoteBook1->Pages;
TabSet1->TabIndex=NoteBook1->PageIndex;
}
//---------------------------------------------------------------------------
```

Активуйте об’єкт **TabSet1** та створіть для нього оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::TabSet1Click(TObject *Sender)
{
NoteBook1->PageIndex=TabSet1->TabIndex;
}
//---------------------------------------------------------------------------
```

Тепер кожне клацання мишею на укладці буде призводити до появи відповідної сторінки блокнота.

Для того, щоб вміст рядка редагування **EditItem** додавався до активної сторінки блокнота, активуйте кнопку **Add** та створіть для неї оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TForm1::ButAddClick(TObject *Sender)
{
if(EditItem –> Text != "")
{
switch(NoteBook1 -> PageIndex)
{
case 0: LstJan->Items->Add ( EditItem->Text); break;
case 1: LstFeb->Items->Add ( EditItem->Text); break;
case 2: LstMar->Items->Add ( EditItem->Text); break;
case 3:  LstApr->Items->Add( EditItem->Text); break;
case 4: LstMay->Items->Add ( EditItem->Text); break;
case 5: LstJun->Items->Add ( EditItem->Text); break;
case 6: LstJul->Items->Add ( EditItem->Text); break;
case 7: LstAug->Items->Add ( EditItem->Text); break;
case 8: LstSep->Items->Add ( EditItem->Text); break;
case 9: LstOct->Items->Add ( EditItem->Text); break;
case 10: LstNow->Items->Add ( EditItem->Text); break;
case 11: LstDec->Items->Add ( EditItem->Text); break; } } }
```

Збережіть проект. Запустіть проект на виконання. Вікно Вашого проекту матиме вигляд як на рис. 3.6:

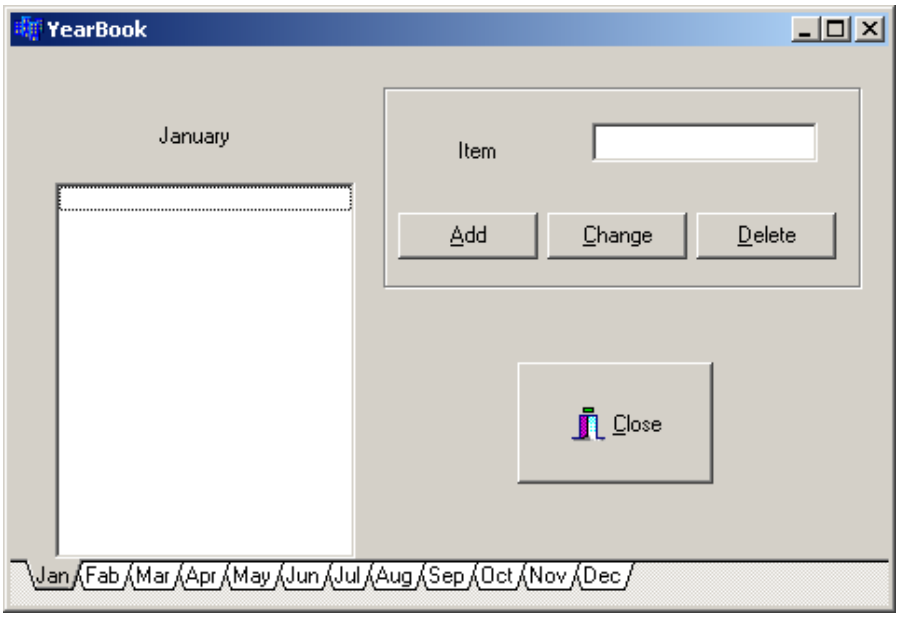


Рис. 3.6. Вікно програмного додатку проекту 3.3

**II. Контрольні завдання**

1. Додати кнопку для отримання списку без рамки.
2. Додати кнопку для відображення списку у декілька колонок.
3. Додати другий список та кнопку. При натисканні на кнопку у другий список копіюється виділений рядок першого списку.
4. Додати другий список. Передбачити діалог при додаванні елементу до другого списку: „Додати рядок до другого списку?”
5. Додати другий список та кнопку. При натисканні на кнопку у другий список копіюються всі виділені рядки першого списку.
6. Реалізувати ситуацію: при заміні виділеного рядка елементи списку знову сортуються.
7. Додати кнопку. При натисканні на кнопку міняються місцями перший та останній рядки списку.
8. Додати кнопку. При натисканні на кнопку з’являється діалогове вікно з текстом виділеного рядка.
9. Додати кнопку. При натисканні на кнопку з’являється діалогове вікно, що відображає почергово рядки списку
10. Додати кнопку. При натисканні на кнопку зі списку видаляються всі виділені рядки.
11. Додати другий список та кнопку. При натисканні на кнопку перший список копіюється до другого.
12. При створенні форми список заповнюється десятьма рядками: „Рядок 1”, „Рядок 2” і т. д.

13. Додати кнопку. При натисканні на кнопку виділений рядок стає першим.
14. Додати кнопку. При натисканні на кнопку список впорядковується за алфавітом.
15. Реалізувати відображення у заголовку форми кількості записів на сторінці.
16. Реалізувати копіювання всіх записів з однієї сторінки на іншу.
17. Реалізувати обробник події клавіатури так, щоб при натисненні на клавішу Delete видалявся виділений запис.
18. Реалізувати переміщення (вирізання-вставку) одного запису з однієї сторінки на іншу.
19. Реалізувати переміщення (вирізання-вставку) всіх записів з однієї сторінки на іншу.
20. Додати кнопку зміни кольору сторінки.
21. Додати кнопку збереження вмісту сторінки у файл.
22. Додати кнопку завантаження вмісту сторінки із файлу.
23. Додати кнопку сортування списку сторінки.
24. Додати можливість пошуку запису у списку сторінки.
25. Додати кнопку відображення у назві закладки кількості записів на сторінці.

### III. Контрольні питання

1. Поясніть призначення компонента **ListBox**. Назвіть його основні властивості.
2. Поясніть призначення компонента **Notebook**. Назвіть його основні властивості.
3. Поясніть призначення компонента **Edit**. Назвіть його основні властивості.
4. Поясніть призначення компонента **TabSet**. Назвіть його основні властивості.
5. Поясніть призначення компонента **GroupBox**. Назвіть його основні властивості.
6. Поясніть призначення компонента **SpinEdit**. Назвіть його основні властивості.
7. Поясніть призначення компонента **Bevel**. Назвіть його основні властивості.
8. Яка властивість компонента ListBox забезпечує роботу з рядками?
9. Назвіть основні методи класу TStrings для роботи з рядками.
10. Яке призначення методу **SaveToFile**()?

11. Яке призначення методу **LoadFromFile**()?

**IV. Рекомендована література:**


1. Архангельский А.Я. C++Builder 6. Справочное пособие. Книга 2. Классы и компоненты / А.Я. Архангельский – М.: Бином-Пресс, 2002. – 528 с.: ил.
2. Кент Рейсдорф, Кен Хендерсон Borland C++Builder. Освой самостоятельно – М.: «Издательство БИНОМ», 1998. – С. 253-273.
3. Сван Т. Основи программирования в Delphi для Windows 95 / Т. Сван. – К. : «Диалектика», 1996. –480 с., ил. – С. 162-189.


## КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 4

**Тема: Конструювання меню**
**Мета:** Розробка додатків зі стандартним та спливаючим меню

**I. Хід виконання роботи.**

Створіть на диску **E:**\ у своєму каталозі новий проект та збережіть його: модулю **Unit1.cpp** надайте ім'я **DialMain.cpp**, а файлу проекту **Project1.bpr** – ім'я **Dial.bpr**.

Помістіть на форму шість позначок **Label** (укладка **Standard**), компонент **PopupMenu** (укладка **Standard**) та компонент **MainMenu** (укладка **Standard**).

Розташування компонентів показано на рис. 4.1:

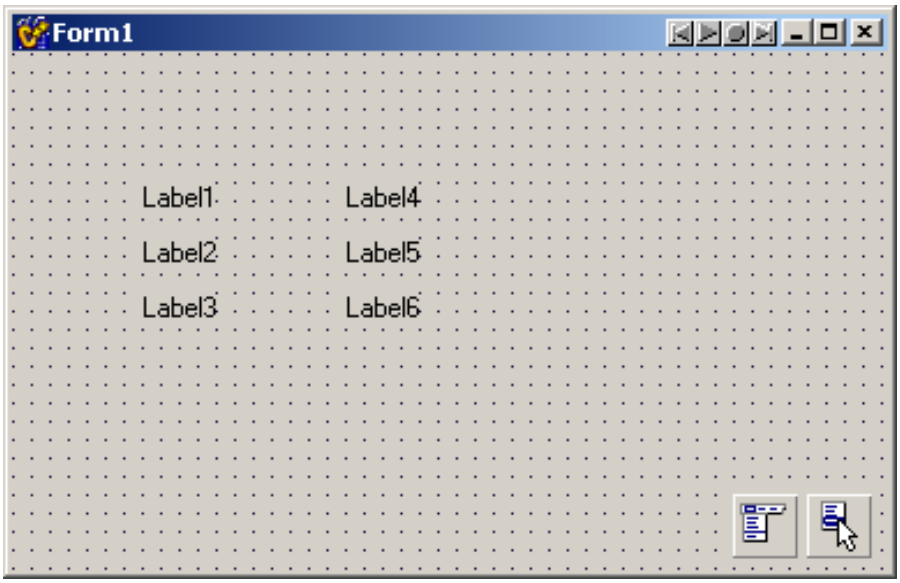


Рис. 4.1. Розташування компонентів проекту 4

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form1 | MainForm | Caption | Dialog |
| | | PopupMenu | PopupMenu1 |
| Label1 | Label1 | Caption | Factor1: |
| Label2 | Label2 | Caption | Factor2: |
| Label3 | Label3 | Caption | Result: |
| Label4 | Factor1Label | Caption | |
| Label5 | Factor2Label | Caption | |
| Label6 | ResultLabel | Caption | |
| PopupMenu1 | PopupMenu1 | | |
| MainMenu1 | DialMainMenu | | |

Кожен елемент (пункт) головного або спливаючого меню – це об’єкт класу **TMenuItem**. Проте, компонент **MenuItem** не входить до палітри VCL. Такі об’єкти найпростіше створювати за допомогою дизайнеру меню **Menu Designer** відповідного компонента меню.

Створіть головне меню додатку. Для цього клацніть двічі на компоненті **DialMainMenu**. В результаті з’явиться вікно дизайнеру меню **Menu Designer (MainForm->DialMainMenu)**. Встановіть імена та властивості для кожного пункту меню згідно таблиці:

| **Об’єкт** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| TMenuItem | FileMenu | Caption | &File |
| | | ShortCut | Ctrl+F |
| TMenuItem | ExitFile | Caption | &Exit |
| | | ShortCut | Ctrl+E |
| TMenuItem | WindMenu | Caption | F&orm |
| | | ShortCut | Ctrl+O |
| TMenuItem | ShowWind | Caption | &Show |
| | | ShortCut | Ctrl+S |

Символ **&** в значенні властивості Caption використовується для визначення клавіші мнемонічного доступу до відповідного пункту меню. Так, запис **&File** визначає доступ до пункту меню **File** за допомогою комбінації клавіш **<Alt+F>.**

Закрийте **Menu Designer**. В результаті цих дій на формі з’явиться головне меню додатку (рис. 4.2):

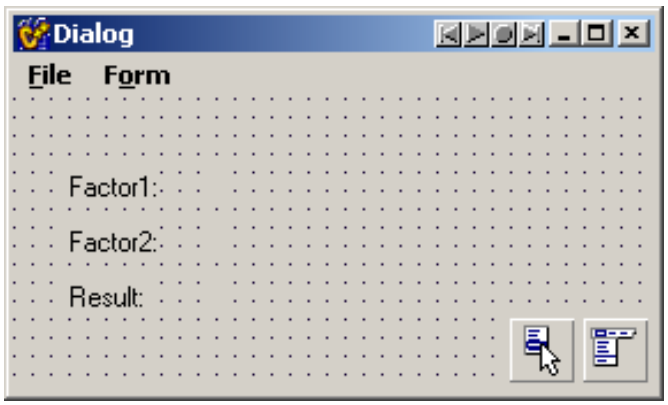


Рис. 4.2. Створення стандартного меню додатку проекту 4

Створіть спливаюче меню. Для цього клацніть двічі на компоненті **PopupMenu1**. В результаті з'явиться вікно Дизайнеру меню **MainForm->PopupMenu1**.

Встановіть імена та властивості кожного пункту меню згідно таблиці:

| **Об'єкт** | **Name** | **Caption** | **Checked** |
|---|---|---|---|
| TMenuItem | RegularFont | Regular Font | true |
| TMenuItem | BoldFont | Bold Font | false |

Командою **File|New->Form** додайте до проекту ще одну форму **Form2.**

Помістіть на форму **Form2** дві позначки **Label** (укладка **Standard)**, два компоненти **Edit** (укладка **Standard)** та дві кнопки **Button** (укладка **Standard).**

Розташування компонентів показано на рис. 4.3:

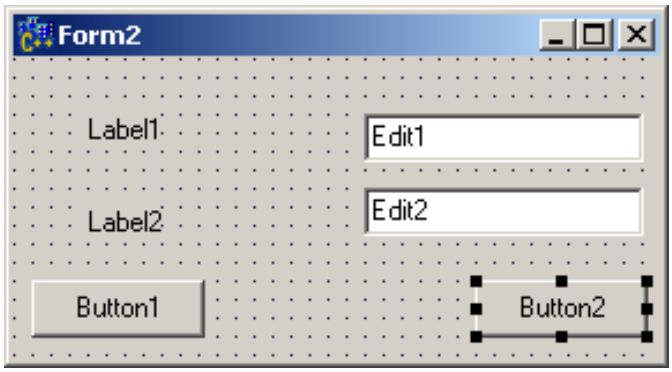


Рис. 4.3. Розташування компонентів підлеглого вікна проекту 4

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form2 | SettingsForm | Caption | Settings |
| Label1 | Label1 | Caption | Factor1: |
| Label2 | Label2 | Caption | Factor2: |
| Button1 | OkButton | Caption | Ok |
| Button2 | CancelButton | Caption | Cancel |
| Edit1 | Factor1Edit | Text | |

| | | | |
|---|---|---|---|
| Edit2 | Factor2Edit | Text | |

Форма матиме вигляд як на рис. 4.4:

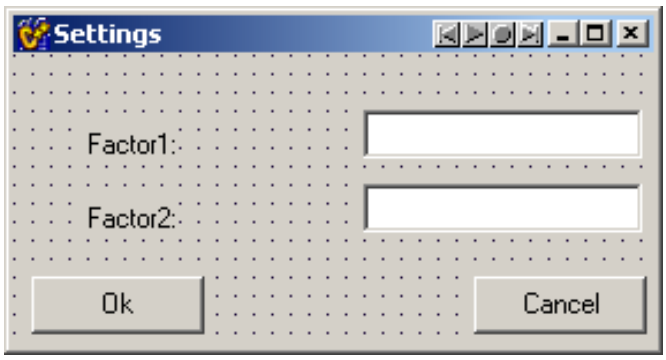


Рис. 4.4. Налаштування властивостей компонентів проекту 4

Збережіть зміни у файл **Unit2.cpp**.

Зверніть увагу, що для взаємодії двох форм між собою вам необхідно до файлу модуля **Unit2.cpp** додати директиву **#include "DialMain.h",** а до файлу модуля **DialMain.cpp** додати директиву **#include "Unit2.h".**

Активуйте форму **Settings** та створіть для неї оброблювач події **OnActivate:**

```
//---------------------------------------------------------------------------
void __fastcall TSettingsForm::FormActivate(TObject *Sender)
{
Factor1Edit->Text=IntToStr(MainForm->Factor1);
Factor2Edit->Text=IntToStr(MainForm->Factor2);
}
//---------------------------------------------------------------------------
```

Створіть оброблювачі події **OnClick** для кнопок **OkButton** та **CancelButton**:

```
//---------------------------------------------------------------------------
void __fastcall TSettingsForm::OkButtonClick(TObject *Sender)
{
MainForm->Factor1Label->Caption=Factor1Edit->Text;
MainForm->Factor2Label->Caption=Factor2Edit->Text;
MainForm->Factor1=StrToInt(Factor1Edit->Text);
MainForm->Factor2=StrToInt(Factor2Edit->Text);
ModalResult=mrOk;
}
//---------------------------------------------------------------------------
void __fastcall TSettingsForm::CancelButtonClick(TObject *Sender)
{
Close();  }
```

```
//---------------------------------------------------------------------------
```

У заголовному файлі **DialMain.h** після директиви public пропишіть змінні, які будуть використані у вашій програмі:

```
//---------------------------------------------------------------------------
public:         // User declarations
        int Factor1, Factor2;
//---------------------------------------------------------------------------
```

Активуйте форму **MainForm** та створіть для неї оброблювач події **OnActivate:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormActivate(TObject *Sender)
{
 Factor1=0;
 Factor2=0;
}
//---------------------------------------------------------------------------
```

Створіть оброблювачі події **OnClick** для пунктів **Exit** та **Show** стандартного меню додатку:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::ExitFileClick(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::ShowWindClick(TObject *Sender)
{
if (SettingsForm->ShowModal() == mrOk)
   {
    ResultLabel->Caption=IntToStr(StrToInt(Factor1Label->Caption) *
StrToInt(Factor2Label->Caption));
   }
}
//---------------------------------------------------------------------------
```

Створіть оброблювачі події **OnClick** для пунктів **Regular Font** та **Bold Font** контекстного меню додатку:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::RegularFontClick(TObject *Sender)
{
ResultLabel->Font->Style=TFontStyles();
```

```
RegularFont->Checked=true;
BoldFont->Checked=false;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::BoldFontClick(TObject *Sender)
{
ResultLabel->Font->Style=TFontStyles()<< fsBold;
RegularFont->Checked=false;
BoldFont->Checked=true;
}
//---------------------------------------------------------------------------
```

Збережіть проект.

Запустіть програму на виконання.

**ІІ. Контрольні завдання**

1. Додайте до стандартного меню пункт для вибору кольору головного вікна з палітри кольорів.
2. Створіть у вторинному вікні головне меню. Передбачте в ньому пункти для вибору варіантів положення цієї форми на екрані.
3. Додайте до контекстного меню пункт для вибору варіантів положення вторинного вікна на екрані.
4. Створіть у вторинному вікні головне меню. Передбачте в ньому пункти для вибору атрибутів шрифту позначок вторинного вікна.
5. Додайте до контекстного меню пункт, у якому можна вибрати шрифт позначок вторинного вікна.
6. Створіть нове вторинне вікно. Додайте до стандартного меню пункт, у якому можна вибрати варіанти заголовку цього вікна.
7. Додайте до стандартного меню пункт, у якому можна вибрати колір позначок вторинного вікна.
8. Створіть у вторинному вікні головне меню. Передбачте в ньому пункт для вибору кольору позначок цього вікна з палітри кольорів.
9. Додайте до контекстного меню пункт, у якому можна вибрати колір позначок вторинного вікна з палітри кольорів.
10. Створіть у вторинному вікні головне меню. Передбачте в ньому пункт для вибору варіантів положення вторинного вікні на екрані.
11. Додайте до стандартного меню пункт для вибору типу (модальне/немодальне) вторинного вікна.
12. Створіть у вторинному вікні контекстне меню. Передбачте в ньому пункт для вибору типу (BorderStyle) вторинного вікна .

13. Додайте до стандартного меню пункт для вибору варіантів розміру головного вікна.
14. Створіть у вторинному вікні контекстне меню. Передбачте в ньому пункт для вибору розміру шрифту позначок вторинного вікна.
15. Додайте до контекстного меню пункт для вибору типу (BorderStyle) вторинного вікна .
16. Створіть у вторинному вікні головне меню. Передбачте в ньому пункт для вибору варіантів розміру вторинного вікна.
17. Додайте до стандартного меню пункт для вибору кольору головного вікні.
18. Створіть у вторинному вікні контекстне меню. Передбачте в ньому пункт для вибору типу (BorderStyle) головної форми.
19. Додайте до контекстного меню пункт для вибору кольору головного вікна.
20. Створіть нове вторинне вікно. Додайте до стандартного меню пункт для виклику цього вікна як немодального.
21. Додайте до контекстного меню пункт для вибору кольору вторинного вікна.
22. Створіть нове вторинне вікно. Додайте до стандартного меню пункт, у якому можна вибрати колір цього вікна з палітри кольорів.
23. Додайте до стандартного меню пункт для вибору кольору вторинного вікна з палітри кольорів.
24. Створіть у вторинному вікні контекстне меню. Передбачте в ньому пункт для вибору типу (BorderStyle) головної форм.
25. Створіть нове вторинне вікно. Додайте до стандартного меню пункт, у якому можна вибрати варіанти розміру цього вікна.

## III. Контрольні питання

1. Поясніть різницю між модальними та немодальними вікнами?
2. Які методи використовуються для виклику модального (немодального) вікна?
3. Яким чином можна визначити клавіші швидкого доступу до пунктів меню?
4. Що визначає властивість **Checked**?

## IV. Рекомендована література

1. Архангельский А.Я. C++Builder 6. Справочное пособие. Книга 2. Классы и компоненты / А.Я. Архангельский – М.: Бином-Пресс, 2002. – 528 с.: ил.

2. Рейсдорф Кент Borland C++Builder. Освой самостоятельно / Кент Рейсдорф, Кен Хендерсон – М.: «Издательство БИНОМ», 1998. –С. 230-249.
3. Сван Т. Основи программирования в Delphi для Windows 95 / Т. Сван. – К. : «Диалектика», 1996. –480 с., ил. – С. 95-118.

## КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 5

**Тема: Використання швидких кнопок**
**Мета:** розробка додатків з реалізацією дій швидких та динамічних кнопок

### I. Хід виконання роботи.

Створіть на диску **E:** у своєму каталозі новий проект та збережіть його: модулю **Unit1.cpp** надайте ім'я **Calc.cpp**, а файлу проекту **Project1.bpr** – ім'я **Calcul.bpr**.

Новий проект передбачає створення калькулятора з використанням компонента **SpeedButton** («швидка» кнопка**)** – кнопки з фіксацією натиснутого стану.

Помістіть на форму два компоненти **Edit (**укладка **Standard)**, один компонент **Bevel (**укладка **Additional)**, одну позначку **Label** (укладка **Standard**) та вісім компонентів **SpeedButton** (укладка **Additional**).

Розташування компонентів показано на рис. 5.1:

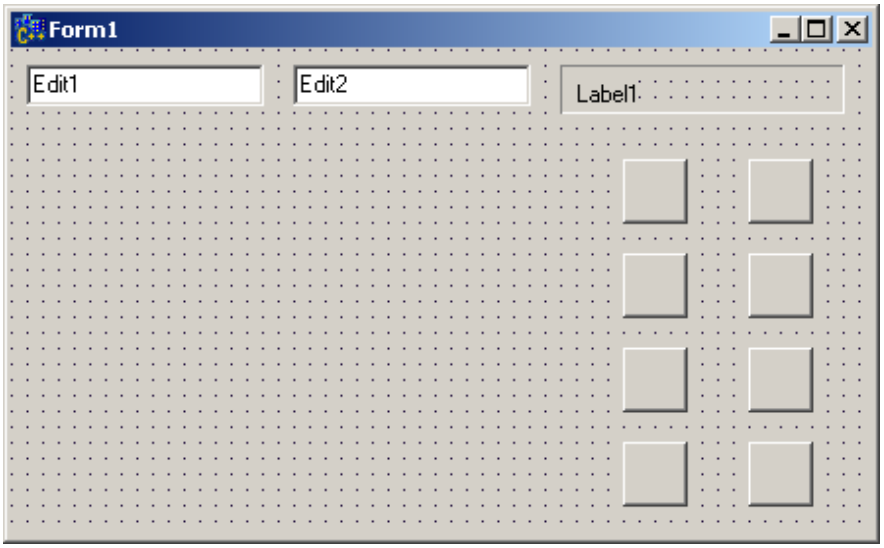


Рис. 5.1. Розташування компонентів проекту 5

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form1 | MainForm | Caption | Calculator |
| | | BorderStyle | bsDialog |
| Edit1 | Edit1 | Text | 0 |

| | | | |
|---|---|---|---|
| | | MaxLength | 16 |
| Edit2 | Edit2 | Text | 0 |
| | | MaxLength | 16 |
| Label1 | Label1 | Caption | 0 |
| SpeedButton1 | Multiply | Caption | * |
| | | Font->Size | 14 |
| SpeedButton2 | Divide | Caption | / |
| | | Font->Size | 14 |
| SpeedButton3 | Plus | Caption | + |
| | | Font->Size | 14 |
| SpeedButton4 | Minus | Caption | – |
| | | Font->Size | 14 |
| SpeedButton5 | ReArrange | Caption | Re |
| SpeedButton6 | Ansver | Caption | Ans |
| SpeedButton7 | Clear | Caption | C |
| SpeedButton8 | Off | Caption | Off |

Форма матиме вигляд як на рис. 5.2:

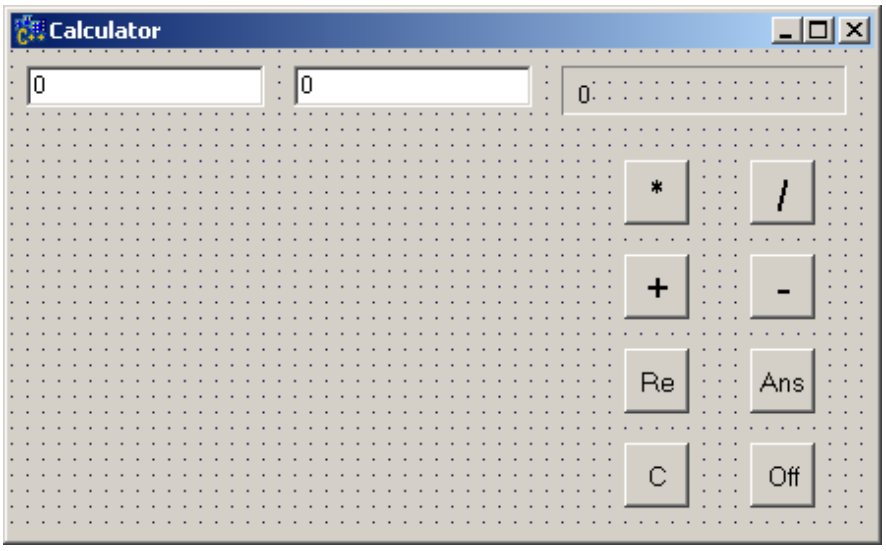


Рис. 5.2. Налаштування властивостей компонентів проекту 5

Збережіть проект.

Швидкі командні кнопки, як правило, використовуються групами. Їх можна вибирати з палітри компонентів (див. вище) або створювати динамічно в процесі розробки програми.

У заголовному файлі **Calc.h** після директиви **private** пропишіть змінні та функції, які будуть використані у вашій програмі:

```
//---------------------------------------------------------------------------
private:        // User declarations
    Extended x, y, Res;
    TSpeedButton *But;
```

```
        void __fastcall CreateButtonKey(AnsiString Cap, int L, int T, int
W, int H, int Key);
        void __fastcall ClickButtonKey(TObject *Sender);
        bool __fastcall GetXY(void);
        void __fastcall ShowRes(void);
    //---------------------------------------------------------------------------
```

У файл модуля **Calc.cpp** додайте опис оголошених функцій:

```
    //---------------------------------------------------------------------------
    void __fastcall TMainForm::CreateButtonKey(AnsiString Cap, int L,
int T, int W, int H, int Key)
    {
    But= new TSpeedButton(this);
    But->Parent=MainForm;
    But->Caption=Cap;
    But->Left=L;
    But->Top=T;
    But->Width=W;
    But->Height=H;
    But->Tag=Key;
    But->AllowAllUp=true;
    But->OnClick=ClickButtonKey;
    }
    //---------------------------------------------------------------------------
    void __fastcall TMainForm::ClickButtonKey(TObject *Sender)
    {
    ActiveControl->Perform(WM_CHAR,
    dynamic_cast<TSpeedButton *>(Sender)->Tag, 0);
    }
    //---------------------------------------------------------------------------
    bool __fastcall TMainForm::GetXY(void)
    {
    if (Edit1->GetTextLen() == 0)
    Edit1->Text = "0";
    if (Edit2->GetTextLen() == 0)
    Edit2->Text = "0";
    try
      {
        x=StrToFloat(Edit1->Text);
        y=StrToFloat(Edit2->Text);
```

```
  }
catch(...)
  {
     MessageDlg("Error!", mtError,  TMsgDlgButtons() << mbOK, 0);
  }
return true;
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::ShowRes(void)
{
Label1->Caption=FloatToStrF(Res, ffGeneral, 15, 15);
}
//---------------------------------------------------------------------------
```

Активуйте форму **MainForm** та створіть для неї оброблювач події **OnCreate**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
x=0;
y=0;
Res=0;
CreateButtonKey("0", 50, 184, 36, 36, 48);
CreateButtonKey(",", 100, 184, 36, 36, 44);
CreateButtonKey("B", 150, 184, 36, 36, 8);
CreateButtonKey("1", 50, 136, 36, 36, 49);
CreateButtonKey("2", 100, 136, 36, 36, 50);
CreateButtonKey("3", 150, 136, 36, 36, 51);
CreateButtonKey("4", 50, 88, 36, 36, 52);
CreateButtonKey("5", 100, 88, 36, 36, 53);
CreateButtonKey("6", 150, 88, 36, 36, 54);
CreateButtonKey("7", 50, 40, 36, 36, 55);
CreateButtonKey("8", 100, 40, 36, 36, 56);
CreateButtonKey("9", 150, 40, 36, 36, 57);
}
//---------------------------------------------------------------------------
```

Активуйте кнопку **Multiply** та створіть для неї оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::MultiplyClick(TObject *Sender)
```

```
{
if (GetXY())
try
  {
    Res=x*y;
    ShowRes();
  }
catch(...)
  {
   MessageDlg("Помилка переповнення!", mtError,
   TMsgDlgButtons() << mbOK, 0);
  }
}
//---------------------------------------------------------------------------
```

Активуйте кнопку **Divide** та створіть для неї оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::DivideClick(TObject *Sender)
{
if (GetXY())
try
  {
     Res=x/y;
     ShowRes();
  }
catch(...)
  {
      MessageDlg("Ділення на нуль!", mtError,  TMsgDlgButtons()
      << mbOK, 0);
  }
}
//---------------------------------------------------------------------------
```

Активуйте кнопку **Plus** та створіть для неї оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::PlusClick(TObject *Sender)
{
if (GetXY())
try
  {
```

```
      Res=x+y;
      ShowRes();
    }
  catch(...)
    {
     MessageDlg("Помилка переповнення!", mtError,
     TMsgDlgButtons() <<mbOK, 0);
    }
  }
  //---------------------------------------------------------------------------
```

Активуйте кнопку **Minus** та створіть для неї оброблювач події **OnClick:**

```
  //---------------------------------------------------------------------------
  void __fastcall TMainForm::MinusClick(TObject *Sender)
  {
  if (GetXY())
  try
    {
      Res=x-y;
      ShowRes();
    }
  catch(...)
    {
     MessageDlg("Помилка переповнення!", mtError,
     TMsgDlgButtons() <<mbOK, 0);
    }
  }
  //---------------------------------------------------------------------------
```

Активуйте кнопку **ReArrange** та створіть для неї оброблювач події **OnClick:**

```
  //---------------------------------------------------------------------------
  void __fastcall TMainForm::ReArrangeClick(TObject *Sender)
  {
  AnsiString s;
  s=Edit1->Text;
  Edit1->Text=Edit2->Text;
  Edit2->Text=s;
  }
  //---------------------------------------------------------------------------
```

Активуйте кнопку **Answer** та створіть для неї оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::AnswerClick(TObject *Sender)
{
dynamic_cast<TEdit *>(ActiveControl)->Text=FloatToStr(Res);
}
//---------------------------------------------------------------------------
```

Активуйте кнопку **Clear** та створіть для неї оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::ClearClick(TObject *Sender)
{
dynamic_cast<TEdit *>(ActiveControl)->Text="0";
}
//---------------------------------------------------------------------------
```

Активуйте кнопку **Off** та створіть для неї оброблювач події **OnClick:**

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::OffClick(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
```

Активуйте форму **MainForm** та створіть для неї оброблювачі подій **OnKeyDown** та **OnKeyUp**:

```
void __fastcall TMainForm::FormKeyDown(TObject *Sender, WORD
&Key, TShiftState Shift)
{
for (int i = ComponentCount - 1; i >= 0; i--)
 {
  if (dynamic_cast<TSpeedButton *>(Components[i]) != NULL)
    {
     if (dynamic_cast<TSpeedButton *>(Components[i])->Tag == Key |
    dynamic_cast<TSpeedButton *>(Components[i])->Tag == Key-48)
       {
        dynamic_cast<TSpeedButton *>(Components[i])->GroupIndex =
Key;
        dynamic_cast<TSpeedButton *>(Components[i])->Down = true;
       }
    }
```

```
 }
}
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormKeyUp(TObject *Sender, WORD &Key,
                                     TShiftState Shift)
{
for (int i = ComponentCount - 1; i >= 0; i--)
 {
 if (dynamic_cast<TSpeedButton *>(Components[i]) != NULL)
  {
    if (dynamic_cast<TSpeedButton *>(Components[i])->Tag == Key |
   dynamic_cast<TSpeedButton *>(Components[i])->Tag == Key-48)
       {
        dynamic_cast<TSpeedButton*>(Components[i])->GroupIndex=0;
       dynamic_cast<TSpeedButton *>(Components[i])->Down = false;
      } } } }
//---------------------------------------------------------------------------
```

Збережіть проект. Запустіть програму на виконання. Вікно Вашого проекту матиме вигляд як на рис. 5.3:

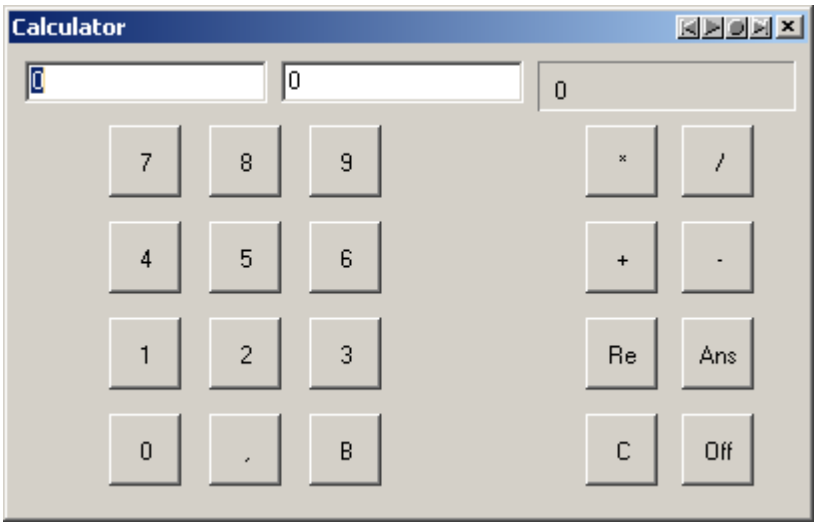


Рис. 5.3. Вікно програмного додатку проекту 5

**II. Контрольні завдання**

1. Додайте кнопку, яка буде реалізовувати обчислення функції cos() від аргумента в активному полі.
2. Додайте кнопку, яка буде реалізовувати обчислення функції sin() від аргумента в активному полі.
3. Додайте кнопку, яка буде реалізовувати обчислення функції ln() від аргумента в активному полі.
4. Додайте кнопку, яка буде реалізовувати обчислення функції sqrt() від

аргумента в активному полі.

5. Додайте кнопку, яка буде реалізовувати обчислення функції x! від аргумента в активному полі.
6. Додайте кнопку, яка буде реалізовувати обчислення функції $x^3$ від аргумента в активному полі.
7. Додайте кнопку, яка буде реалізовувати обчислення функції $x^2$ від аргумента в активному полі.
8. Додайте кнопку, яка буде реалізовувати обчислення функції $10^x$ від аргумента в активному полі.
9. Додайте кнопку, яка буде реалізовувати обчислення функції exp() від аргумента в активному полі.
10. Додайте кнопку, яка буде реалізовувати обчислення функції abs() від аргумента в активному полі.
11. Додайте кнопку, яка буде реалізовувати обчислення функції 1/x від аргумента в активному полі.
12. Додайте кнопку з константою pi, при натисненні на яку в активне поле вводиться відповідне значення.
13. Додайте кнопку з константою e, при натисненні на яку в активне поле вводиться відповідне значення.
14. Додайте кнопку, яка буде реалізовувати обчислення функції a^b.
15. Додайте кнопку, яка буде реалізовувати обчислення функції $\log_a$.
16. Додайте кнопку, яка буде реалізовувати конвертер величин: м–>км.
17. Додайте кнопку, яка буде реалізовувати конвертер величин: г->кг.
18. Додайте кнопку, яка буде реалізовувати розділення розрядів чисел (наприклад, 12345->12 345).
19. Додайте кнопку, яка буде реалізовувати переведення в інші системи числення: двійкова–>десяткова.
20. Додайте кнопку, яка буде реалізовувати переведення в інші системи числення: десяткова –>двійкова.
21. Додайте (динамічно) кнопки для введення щістнадцяткових чисел (A, B, C, D, E, F).
22. Змініть код програми таким чином, щоб при введенні даних в перший елемент редагування, вони одночасно з’являлись й у другому елементі.
23. Додайте список та кнопки запису у пам'ять та зчитування. При натисненні на запис до списку додається значення змінної, при зчитування змінній присвоюється значення вибраного значення зі списку.
24. Реалізуйте операції роботи с пам’яттю ( MC, MR, MS ).
25. Додайте список у якому будуть зберігатись та відображатись виконані

дії (наприклад, 1+2=3 ).

**III. Контрольні питання**

1. У чому полягає відмінність кнопок **SpeedButton** та **Button**?
2. Як можна об’єднати кілька кнопок **SpeedButton** в одну групу?
3. Який клас призначений для обробки виняткових ситуацій?
4. Назвіть основні властивості винятків. Поясніть їх використання для збільшення надійності програми.

**IV. Рекомендована література**


1. Архангельский А.Я. C++Builder 6. Справочное пособие. Книга 2. Классы и компоненты / А.Я. Архангельский – М.: Бином-Пресс, 2002. – 528 с.: ил.
2. Сван Т. Основи программирования в Delphi для Windows 95 / Т. Сван. – К. : «Диалектика», 1996. – С. 119-138.


## КОМП’ЮТЕРНИЙ ПРАКТИКУМ № 6

**Тема: Створення інструментальних панелей**
**Мета:** розробка додатків з панелями інструментів та рядків стану

**I. Хід виконання роботи.**

Створіть на диску **E:**\ у своєму каталозі новий проект та збережіть його: імена модуля **Unit1.cpp** та файлу проекту **Project1.bpr** залиште за замовчуванням.

Новий проект передбачає створення текстового редактора на основі багаторядкового вікна редагування **RichEdit**. Вікно текстового редактора міститиме: головне меню, панель інструментів, що дублюватиме основні розділи меню, та рядок стану.

Помістіть на форму компоненти **RichEdit (**укладка **Win32), StatusBar (**укладка **Win32), ToolBar** (укладка **Win32), ImageList (**укладка **Win32**), **ActionList (**укладка **Standard**) та **MainMenu** (укладка **Standard).**

Встановіть властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивості** | **Значення** |
|---|---|---|---|
| Form1 | Form1 | Caption | Text Editor |
| | | Width | 500 |
| | | Height | 350 |
| RichEdit1 | RichEdit1 | Align | alClient |
| | | Lines | |

| | | | |
|---|---|---|---|
| MainMenu1 | MainMenu1 | Images | ImageList1 |
| ToolBar1 | ToolBar1 | Indent | 4 |
| | | Images | ImageList1 |
| | | ShowHint | true |
| ActionList1 | ActionList1 | Images | ImageList1 |
| ImageList1 | ImageList1 | | |
| StatusBar1 | StatusBar1 | | |

Форма матиме вигляд як на рис. 1:

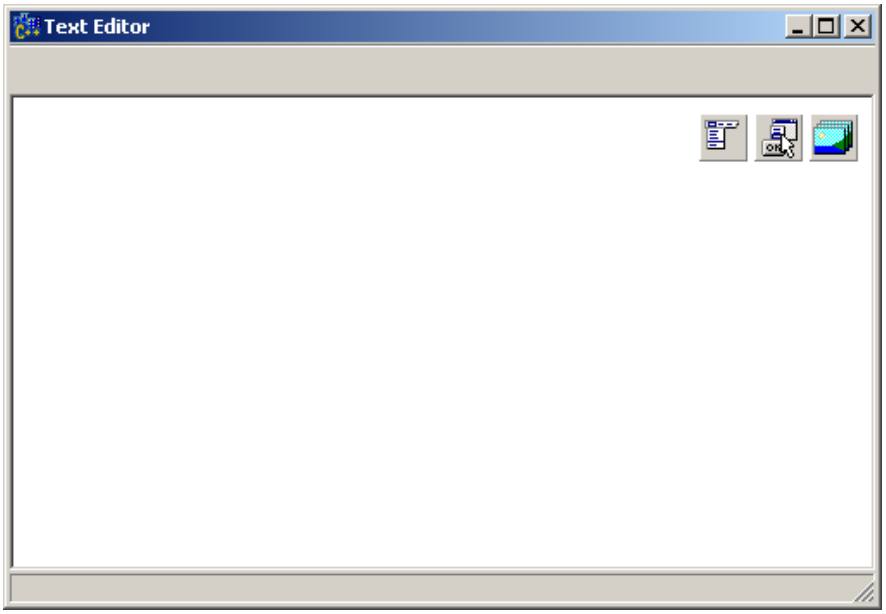


Рис. 6.1. Налаштування властивостей компонентів проекту 6

Компонент **RichEdit** (Текстовий редактор) представляє собою стандартний елемент керування Windows Rich text edit control (Область форматування тексту). Компонент працює з текстом у збагаченому форматі RTF. Основна особливість даного компонента – це можливість форматування окремих абзаців чи фрагментів тексту. У цьому компоненті можна змінювати шрифти, використовувати відступи, задавати напівжирний, курсивний чи підкреслений шрифт та інше. Після розміщення на формі даний компонент готовий до роботи.

Компонент **StatusBar** (Рядок стану) являє собою ряд панелей, що відображають рядок стану в стилі Windows. Зазвичай використовується для виведення додаткової інформації чи підказки. Після розміщення компонента на формі рядок стану автоматично прикріплюється до нижньої частини форми.

Компонент **ToolBar** слугує для створення інструментальних панелей швидких кнопок додатку. Після розміщення компонента на формі він автоматично прикріплюється до верхньої частини форми. Панель зв’язується з компонентом **ImageList,** який являє собою набір піктограм, а в її кнопках

даються посилання на дії, які описані в компоненті **ActionList.**

Компонент **ActionList** (Список дій) слугує для диспетчеризації дій. Він спрощує створення інструментальних панелей додатку.

Методика розробки додатку передбачає виконання послідовності кроків:

1. *Створюємо список дій* текстового редактору, які повинні бути доступні користувачеві через розділи меню та інструментальні панелі: **New**, **Open**, **Save**, **Save As**, **Exit**, **Cut**, **Copy**, **Paste**, **About**.

Необхідні дії до компоненту **ActionList** вводяться та впорядковуються за допомогою Редактора дій, який викликається подвійним клацанням на компоненті **ActionList1**. Активуйте контекстне меню Редактора дій **Editing Form1->ActionList1** правою кнопкою миші у вікні **Actions** та оберіть команду **New Action** (нова дія). У списку з'явиться нова дія **Action1.**

Додайте до списку ще п'ять дій (рис. 6.2):

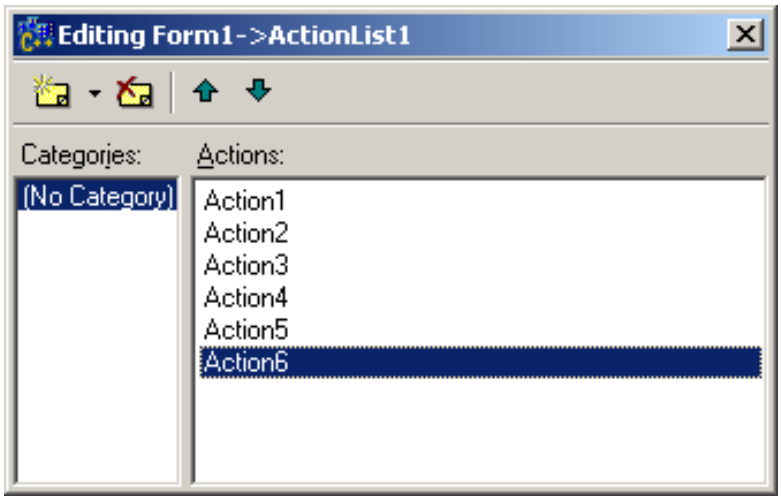


Рис. 6.2. Вікно редактору дій

Кожен елемент списку – це об'єкт класу **TAction**. Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Caption** | **Category** | **Hint** | **ImageIndex** |
|---|---|---|---|---|---|
| Action1 | FileNew | &New | File | Create file. | 0 |
| Action2 | FileOpen | &Open | File | Open file | 1 |
| Action3 | FileSave | &Save | File | Save file | 2 |
| Action4 | FileSaveAs | Save&As | File | Save file as | -1 |
| Action5 | FileExit | E&xit | File | Exit application | 3 |
| Action6 | HelpAbout | &About | Help | Help | -1 |

Тепер додайте до списку стандартні дії: **Cut**, **Copy**, **Paste.** Для цього в контекстному меню Редактора дій оберіть команду **New Standard Action** (нова стандартна дія). Команда **New Standard Action** відкриває вікно **Standard Action Classes**, в якому можна вибрати необхідну стандартну дію.

Оберіть у категорії **Edit** дії **TEditCut**, **TEditCopy** та **TEditPaste.**

Встановіть значення їх властивостей відповідно до таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| TEditCut1 | TEditCut1 | ImageIndex | 4 |
| TEditCopy1 | TEditCopy1 | ImageIndex | 5 |
| TEditPaste1 | TEditPaste1 | ImageIndex | 6 |

Перевага стандартних дій в тім, що для них не потрібно писати обробники подій. Всі операції, необхідні для виконання стандартних дій, вже закладено в їхні об’єкти.

Для нестандартних дій обробники подій потрібно прописувати (див. далі).

Закрийте вікно Редактора дій.

Збережіть проект.

2. *Заповнюємо список зображень* **ImageList1** піктограмами, які будуть використовуватися для тих дій, які повинні бути доступними зі швидких кнопок панелі інструментів (**New**, **Open**, **Save**, **Exit**, **Cut**, **Copy**, **Paste)**.

Необхідні зображення до компоненту **ImageList** вводяться та впорядковуються за допомогою Редактору зображень **Form1->ImageList1 ImageList**, який викликається подвійним клацанням на компоненті **ImageList1**. Клацніть на кнопці **Add** та перейдіть у директорію

**C:\Program Files\Common Files\Borland Shared\Images\Buttons**.

Оберіть файл **filenew.bmp (**ImageIndex=0).

Кожна іконка містить активне та сіре зображення. Вам буде запропоновано розділити зображення на два. Завжди обирайте відповідь **Yes**. Ви побачите обидва зображення. Завжди видаляйте сіре (друге) зображення.

Аналогічно додайте до списку зображень файли: **fldropen.bmp (**ImageIndex=1), **floppy.bmp (**ImageIndex=2**), doorshut.bmp (**ImageIndex=3**), cut.bmp (**ImageIndex=4**), copy.bmp (**ImageIndex=5) та **paste.bmp (**ImageIndex =6) (рис. 6.3):

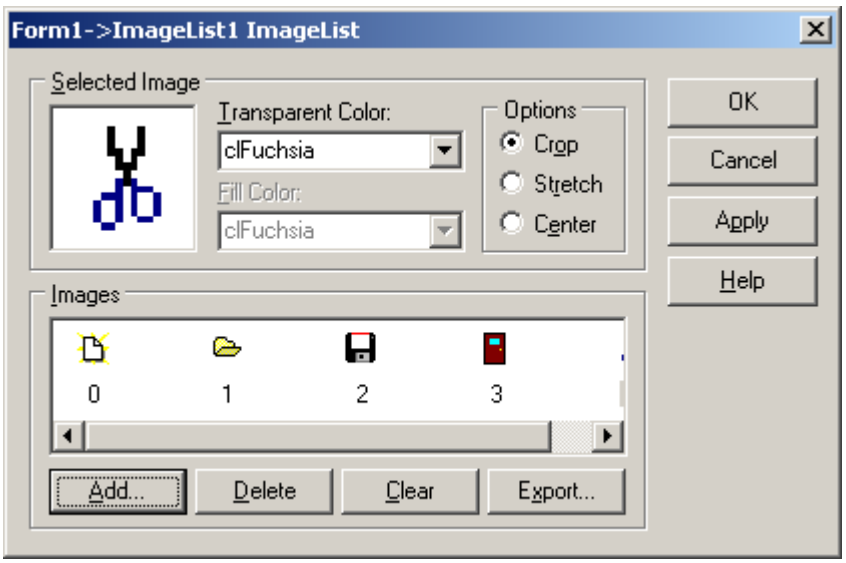


Рис. 6.3. Вікно редактору зображень ImageList

Закрийте редактор **ImageList** кнопкою **OK**.

Перевірити список дій та відповідних зображень можна за допомогою Редактору дій **Editing Form1->ActionList1** (рис. 4):

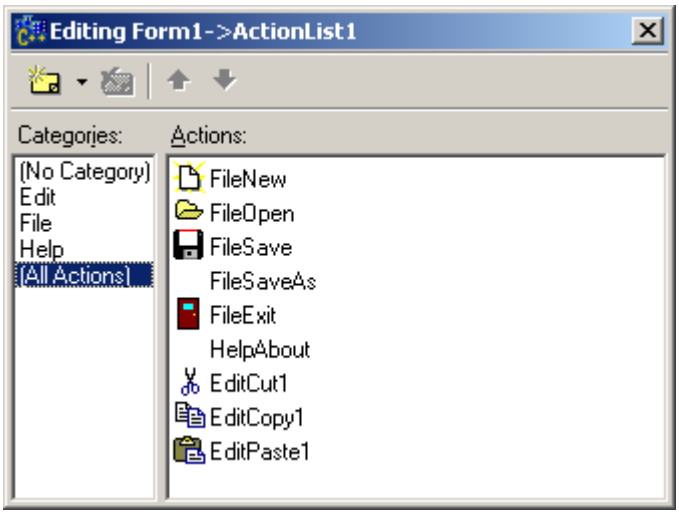


Рис. 6.4. Список дій і відповідних зображень проекту 6

Збережіть проект.

3. *Створюємо головне меню* додатку.

Подвійним клацанням миші на компоненті **MainMenu1** відкрийте Дизайнер меню **Form1->MainMenu1.**

Встановіть імена та властивості пунктів стандартного меню додатку згідно таблиці:

| **Компонент** | **Name** | **Caption** |
|---|---|---|
| TMenuItem | FileMenu | &File |
| TMenuItem | EditMenu | &Edit |
| TMenuItem | HelpMenu | &Help |

Активуйте пункт меню **FileMenu**. Встановіть імена та властивості

команд даного пункту меню згідно таблиці (рис. 5):

| **Компонент** | **Name** | **Caption** | **Action** |
|---|---|---|---|
| TMenuItem | NewFile | &New | FileNew |
| TMenuItem | OpenFile | &Open | FileOpen |
| TMenuItem | N1 | _ | |
| TMenuItem | SaveFile | &Save | FileSave |
| TMenuItem | SaveAsFile | Save&As | FileSaveAs |
| TMenuItem | N2 | _ | |
| TMenuItem | ExitFile | E&xit | FileExit |

Активуйте пункт меню **EditMenu**. Встановіть імена та властивості команд даного пункту меню згідно таблиці:

| **Компонент** | **Name** | **Caption** | **Action** |
|---|---|---|---|
| TMenuItem | CutEdit | Cu&t | EditCut1 |
| TMenuItem | CopyEdit | &Copy | EditCopy1 |
| TMenuItem | PasteEdit | &Paste | EditPaste1 |

Активуйте пункт меню **HelpMenu**. Встановіть ім’я та властивості команди даного пункту меню згідно таблиці

| **Компонент** | **Name** | **Caption** | **Action** |
|---|---|---|---|
| TMenuItem | AboutHelp | A&bout | HelpAbout |

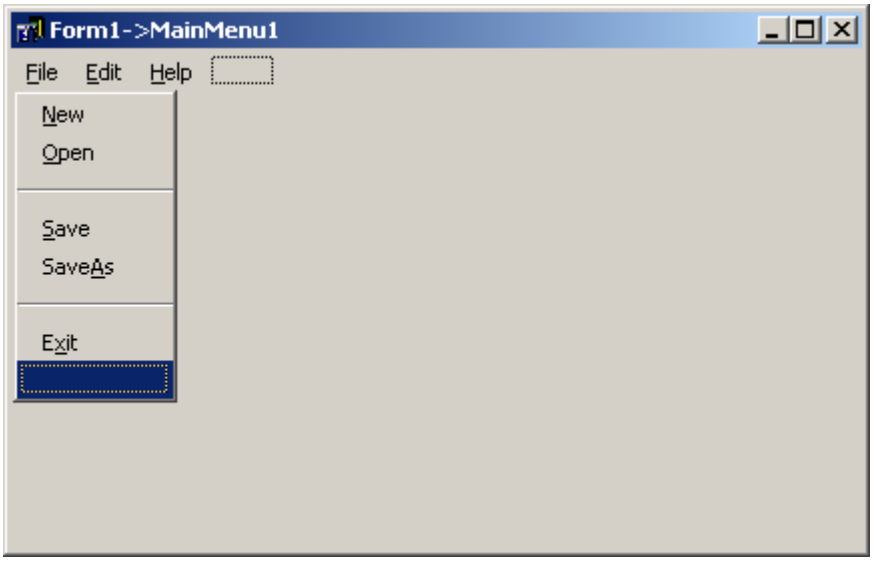


Рис. 6.5. Створення стандартного меню додатку проекту 6

Закрийте Дизайнер меню **Menu Designer Form1->MainMenu1.**
Збережіть проект.
Форма буде матиме вигляд як на рис. 6.6:

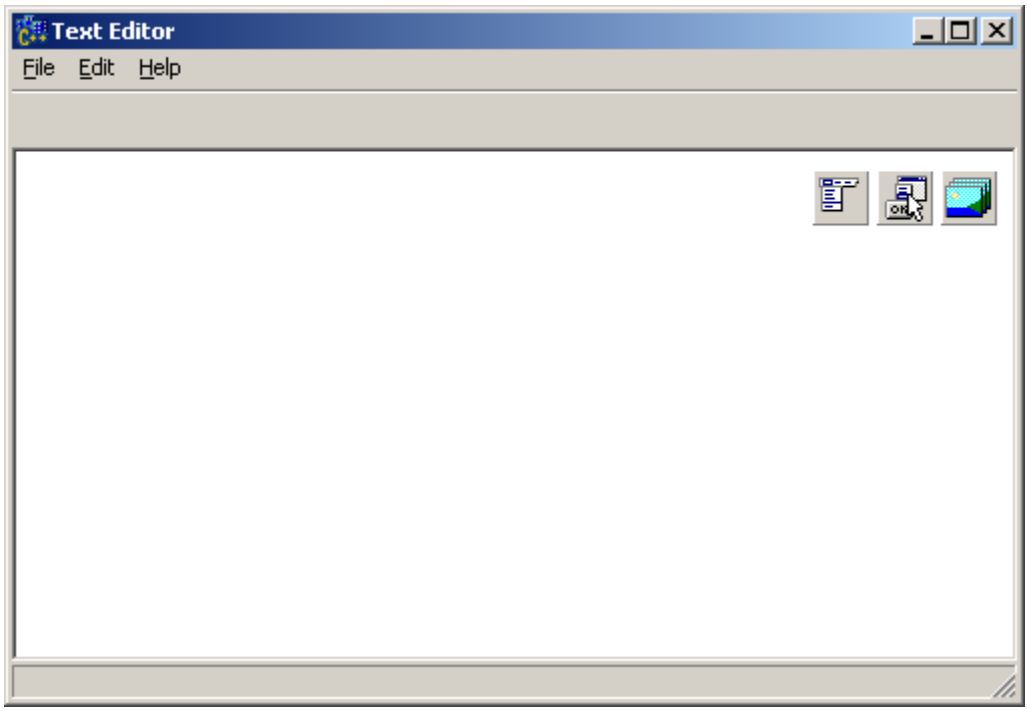


Рис. 6.6. Вікно додатку проекту 6 з рядком головного меню

4. *Створюємо панель інструментів* додатку.

Кнопки та роздільники на панель інструментів додаються за допомогою контекстного меню компонента **ToolBar1.** Оберіть в меню команду **New Button**. На панелі з'явиться кнопка (**New**) – об'єкт класу **TToolButton**. Аналогічно додайте на панель ще три кнопки (**Open, Save, Exit**). Для створення роздільника оберіть в контекстному меню команду **New Separator** – на панелі з'явиться роздільник. Додайте на панель ще три кнопки (**Cut, Copy, Paste**).

Встановіть імена та властивості кнопок згідно таблиці:

| **Компонент** | **Name** | **Action** |
|---|---|---|
| ToolButton1 | ToolButton1 | FileNew |
| ToolButton2 | ToolButton2 | FileOpen |
| ToolButton3 | ToolButton3 | FileSave |
| ToolButton4 | ToolButton4 | FileExit. |
| ToolButton5 | ToolButton5 | |
| ToolButton6 | ToolButton6 | EditCut1 |
| ToolButton7 | ToolButton7 | EditCopy1 |
| ToolButton8 | ToolButton8 | EditPaste1 |

Збережіть проект.

Форма матиме вигляд як на рис. 6.7:

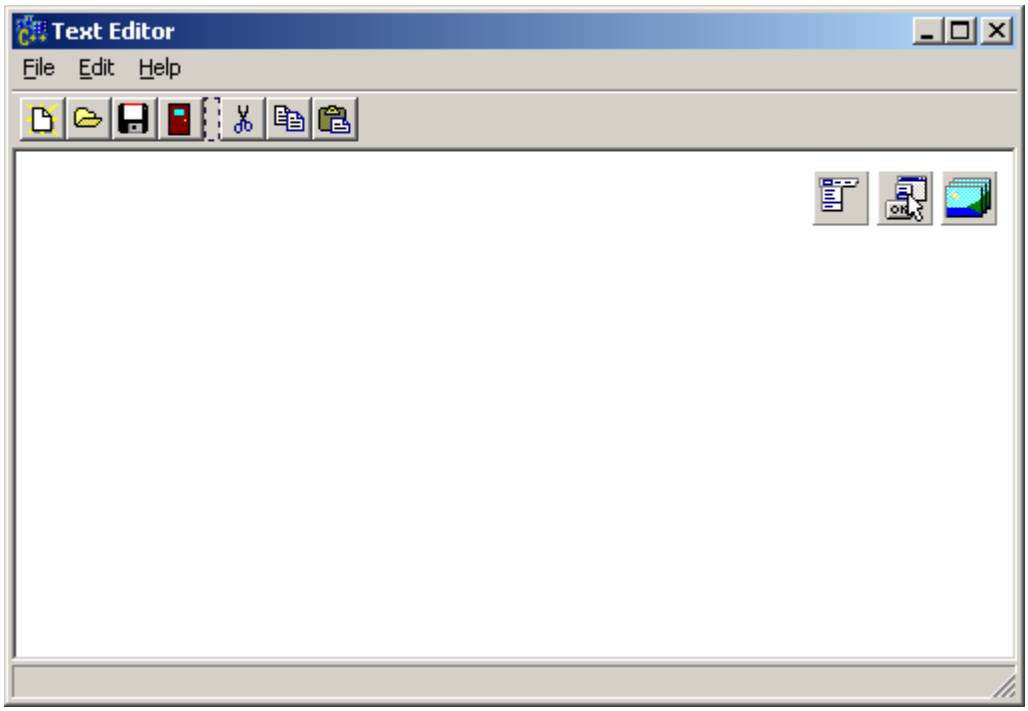


Рис. 6.7. Вікно додатку проекту 6 з панеллю інструментів

5. *Налагоджуємо панель рядка стану* додатку.

Рядок стану може містити одну чи кілька панелей. Кожна панель являє собою об'єкт класу **TStatusPanels**. Властивості панелей задаються за допомогою Редактору наборів **Editing StatusBar1->Panels**, який викликається подвійним клацанням на компоненті **StatusBar1**. Правою кнопкою миші активуйте контекстне меню Редактору та оберіть команду **Add**, щоб додати панель рядка стану (рис. 6.8).

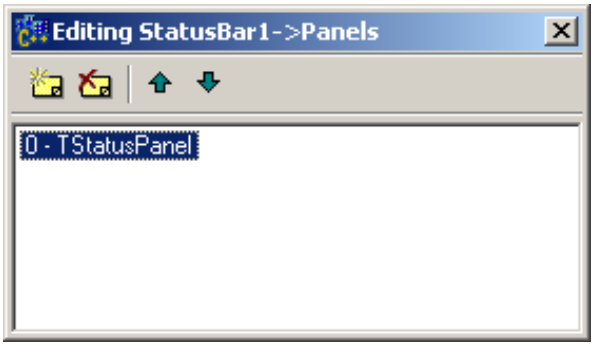


Рис. 6.8. Вікно Редактору наборів

Закрийте Редактор панелей.

6. *Прописуємо обробники подій для нестандартних дій* списку **ActionList1**

У заголовному файлі **Unit1.h** після директиви **public:** пропишіть змінну, яка буде використовуватись у вашій програмі:

```
//---------------------------------------------------------------------------
public:		// User declarations
```

**AnsiString FileName;**

//---------------------------------------------------------------------------

Помістіть на форму компоненти **OpenDialog** (укладка **Dialogs)** та **SaveDialog** (укладка **Dialogs).**

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| OpenDialog1 | OpenDialog1 | DefaultExt | txt |
| | | Title | Open File |
| SaveDialog1 | SaveDialog1 | DefaultExt | txt |
| | | Title | Save As |

Значення властивості **Filter** компонентів **OpenDialog** та **SaveDialog** встановлюється за допомогою Редактора фільтра **Filter Editor**, який викликається подвійним клацанням на властивості **Filter.** Встановіть параметри фільтра згідно таблиці:

| **Filter Name** | **Filter** |
|---|---|
| Text files | *.txt |
| All files | *.* |

Закрийте Редактор фільтра.

Викличте Редактор дій **Editing Form1->ActionList1.**

Активуйте компонент **FileNew** категорії **File** та створіть для нього обробник події **OnExecute**:

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FileNewExecute(TObject *Sender)
{
RichEdit1->Clear();
FileName = "Untitled.txt";
StatusBar1->Panels->Items[0]->Text = FileName;
}
//---------------------------------------------------------------------------
```

Активуйте компонент **FileOpen** та створіть для нього обробник події **OnExecute**:

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FileOpenExecute(TObject *Sender)
{
if (OpenDialog1->Execute())
 {
  RichEdit1->Lines->LoadFromFile(OpenDialog1->FileName);
  FileName = OpenDialog1->FileName;
```

```
  StatusBar1->Panels->Items[0]->Text = FileName;
 }
}
//---------------------------------------------------------------------------
```

Активуйте компонент **FileSave** та створіть для нього обробник події **OnExecute**:

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FileSaveExecute(TObject *Sender)
{
if (FileName == "Untitled.txt")
  FileSaveAsExecute(NULL);
else
  RichEdit1->Lines->SaveToFile(FileName);
}
//---------------------------------------------------------------------------
```

Активуйте компонент **FileSaveAs** та створіть для нього обробник події **OnExecute**:

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FileSaveAsExecute(TObject *Sender)
{
SaveDialog1->FileName = FileName;
SaveDialog1->InitialDir = ExtractFilePath(FileName);
if (SaveDialog1->Execute())
 {
  RichEdit1->Lines->SaveToFile(SaveDialog1->FileName);
  FileName = SaveDialog1->FileName;
  StatusBar1->Panels->Items[0]->Text = FileName;
 }
}
//---------------------------------------------------------------------------
```

Активуйте компонент **FileExit** та створіть для нього обробник події **OnExecute**:

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FileExitExecute(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
```

Закрийте Редактор дій.

Додайте до проекту ще одну форму командою **File|New->Other.** У діалоговому вікні **New Items** на закладці **Forms** оберіть **About Box.** З'явиться заготовка форми.

Встановіть імена та властивості компонентів згідно таблиці:

| **Name** | **Властивість** | **Значення** |
|---|---|---|
| AboutBox | Caption | About Text Editor |
| ProductName | Caption | Text Editor |
| Version | Caption | Version 1.0 |
| Copyright | Caption | Copyright 2009 |
| Comments | Caption | |

Форма матиме вигляд як на рис. 6.9:

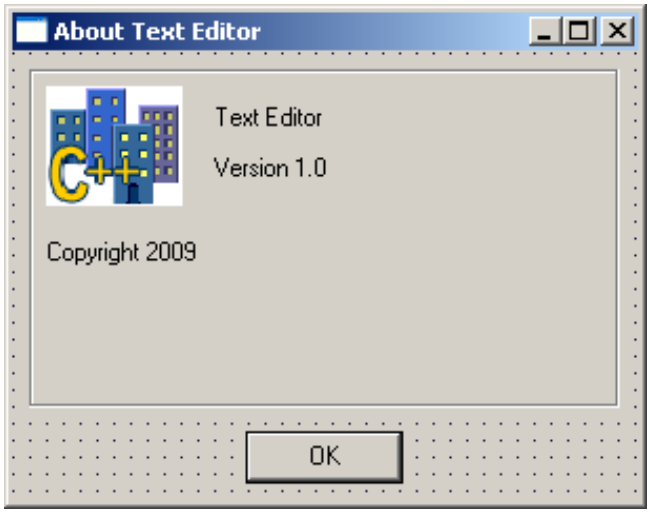


Рис. 6.9. Вигляд підлеглого вікна проекту 6

Збережіть зміни у файл **About.cpp.**

У файл **Unit1.cpp** додайте директиву **#include "About.h".**

Викличте Редактор дій **Editing Form1->ActionList1.** Активуйте компонент **HelpAbout** категорії **Help** та створіть для нього обробник події **OnExecute**:

```
//---------------------------------------------------------------------------
void __fastcall TForm1:: HelpAboutExecute (TObject *Sender)
{
AboutBox->ShowModal();
}
//---------------------------------------------------------------------------
```

Закрийте Редактор дій.

Активуйте головну форму **Form1** та створіть для неї обробник події **OnCreate:**

```
//---------------------------------------------------------------------------
void __fastcall TForm1::FormCreate(TObject *Sender)
{
```

```
FileName = "Untitled.txt";
StatusBar1->Panels->Items[0]->Text = FileName;
RichEdit1->Clear();
}
//---------------------------------------------------------------------------
```

Збережіть проект. Запустіть програму на виконання. Вікно Вашого проекту матиме вигляд як на рис. 6.10 (для прикладу в редакторі відкрито текстовий файл ReadMe.txt):

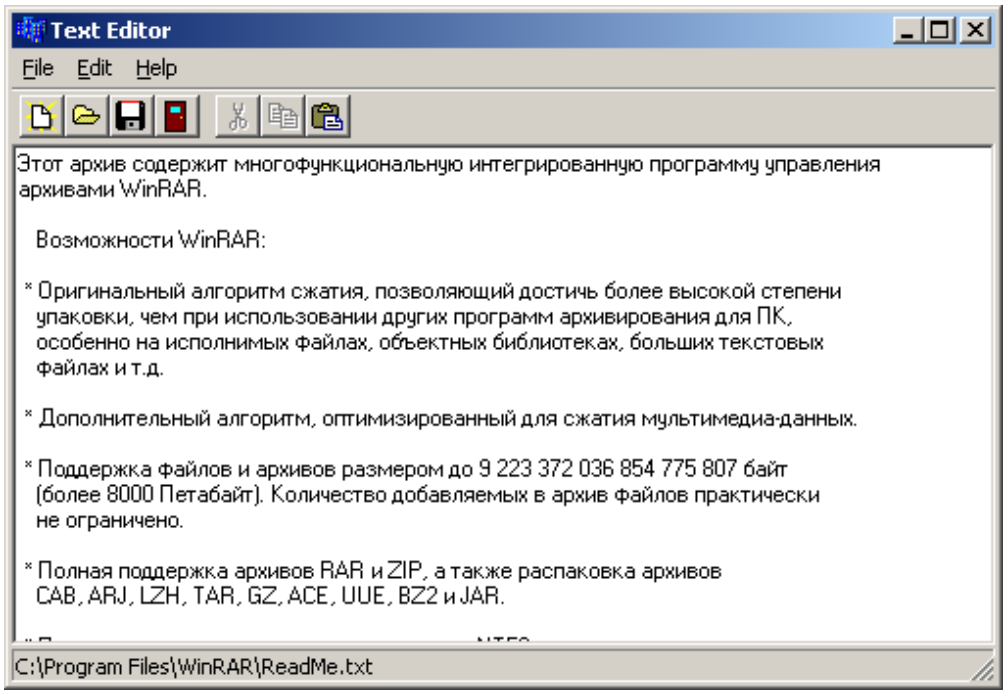


Рис. 6.10. Вікно програмного додатку проекту 6

### III. Контрольні завдання

1. Додайте на панель інструментів кнопки для зміни накреслення шрифту (жирний, курсив, підкреслений ).
2. Додайте на панель інструментів кнопку для виклику діалогового вікна палітри кольорів (для зміни кольору шрифта).
3. Додайте на панель інструментів кнопку для зміни горизонтального положення форми.
4. Додайте на панель інструментів кнопку для додавання нового рядка в кінець тексту.
5. Додайте на панель інструментів кнопку для збільшення абзацного відступу.
6. Додайте на панель інструментів кнопку для вирівнювання виділеного абзацу за правим краєм.
7. Додайте на панель інструментів кнопку для виклику діалогового вікна шрифту.

8. Додайте на панель інструментів кнопку для зміни параметрів шрифту на задані.
9. Додайте на панель інструментів кнопку для видалення з тексту виділеного рядка.
10. Додайте на панель інструментів кнопку для додавання в кінець тексту виділеного рядка.
11. Додайте на панель інструментів кнопку для створення нумерованого списку.
12. Додайте на панель інструментів кнопку для додавання нового рядка на початок тексту.
13. Додайте на панель інструментів кнопку для заміни місцями першого та останнього рядків тексту.
14. Додайте на панель інструментів кнопку для створення маркерованого списку.
15. Додайте на панель інструментів кнопку для вирівнювання виділеного абзацу за центром.
16. Додайте на панель інструментів кнопку для видалення всього тексту.
17. Додайте на панель інструментів кнопку для зміни розміру форми.
18. Додайте на панель інструментів кнопку для вирівнювання виділеного абзацу за шириною.
19. Додайте на панель інструментів кнопку для зміни кольору шрифту на заданий.
20. Додайте на панель інструментів кнопку для зміни вертикального положення форми.
21. Додайте на панель інструментів кнопку для автоматичного збереження файлу.
22. Додайте на панель інструментів кнопку для копіювання виділеного фрагменту тексту у буфер обміну.
23. Додайте на панель інструментів кнопку для вставки тексту з буферу обміну.
24. Додайте на панель інструментів кнопку для пошуку слова у тексті.
25. Додайте на панель інструментів кнопку для реалізації заміни слова у тексті.

### III. Контрольні питання

1. Назвіть основні властивості компонента **RichEdit**?
2. Як використовується компонент **ActionList**?
3. Коли виникає подія **OnExecute**?
4. Яке призначення методу **Execute**()?

**IV. Рекомендована література**


1. Архангельский А.Я. C++Builder 6. Справочное пособие. Книга 2. Классы и компоненты / А.Я. Архангельский – М.: Бином-Пресс, 2002. – 528 с.: ил.
2. Рейсдорф Кент Borland C++Builder. Освой самостоятельно / Кент Рейсдорф, Кен Хендерсон – М.: «Издательство БИНОМ», 1998. –С. 423-436.
3. Сван Т. Основи программирования в Delphi для Windows 95 / Т. Сван. – К. : «Диалектика», 1996. – С. 139-161.


## КОМП'ЮТЕРНИЙ ПРАКТИКУМ № 7

**Тема:. Розробка додатків ведення баз даних Розробка**
**Мета:** розробка систем управління базами даних

**І. Хід виконання роботи.**

**Проект 7.1.**

Створіть на диску **E:**\ у своєму каталозі новий проект та збережіть його: модулю **Unit1.cpp** надайте ім'я **Baza.cpp**, а файлу проекту **Project1.bpr** – ім'я **BDan.bpr**.

Новий проект передбачає створення системи керування базою даних. Вікно додатка міститиме: головне меню, панель інструментів та рядок стану.

Помістіть на форму компоненти **StatusBar (**укладка **Win32), ToolBar** (укладка **Win32), Panel** (укладка **Standard), ImageList (**укладка **Win32**), **ActionList (**укладка **Standard**) та **MainMenu** (укладка **Standard).**

Встановіть властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| TForm1 | TForm1 | Caption | Biolife |
| | | Width | 630 |
| StatusBar1 | StatusBar1 | AutoHint | true |
| ImageList1 | ImageList1 | | |
| ActionList1 | ActionList1 | Images | ImageList1 |
| ToolBar1 | ToolBar1 | Indent | 4 |
| | | Images | ImageList1 |
| | | ShowHint | true |
| MainMenu1 | MainMenu1 | Images | ImageList1 |
| Panel1 | Panel1 | Align | alTop |

| | | | |
|---|---|---|---|
| | | Height | 180 |
| | | Caption | |
| | | Color | clBackground |

Форма матиме вигляд як на рис. 7.1:

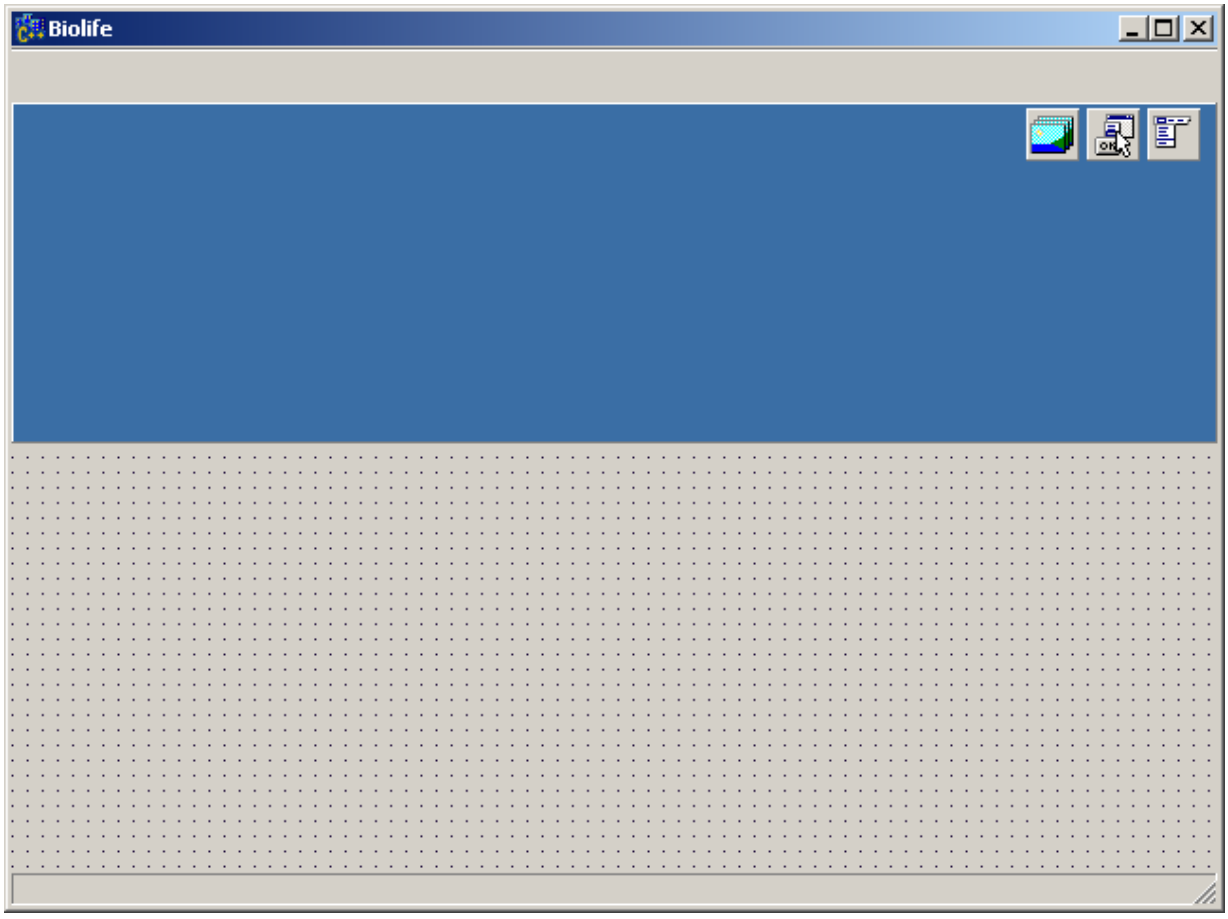


Рис. 7.1. Налаштування властивостей компонентів проекту 7.1

Помістіть на форму компоненти баз даних: **Table (**укладка **BDE), DataSource (**укладка **Data Access)** та **DBGrid (**укладка **Data Controls).**

Встановіть властивості компонентів згідно таблиці:

| | | | |
|---|---|---|---|
| Table1 | Table1 | DatebaseName | BCDEMOS |
| | | TableName | biolife.db |
| | | ReadOnly | true |
| | | Active | true |
| DataSource1 | DataSource1 | DataSet | Table1 |
| DBGrid1 | DBGrid1 | Align | alTop |
| | | Height | 225 |
| | | DataSource | DataSource1 |
| | | Anchors | akLeft = true |
| | | | akTop = true |
| | | | akRight = true |

| | | | |
|---|---|---|---|
| | | | akBottom = true |

Форма матиме вигляд як на рис. 7.2:

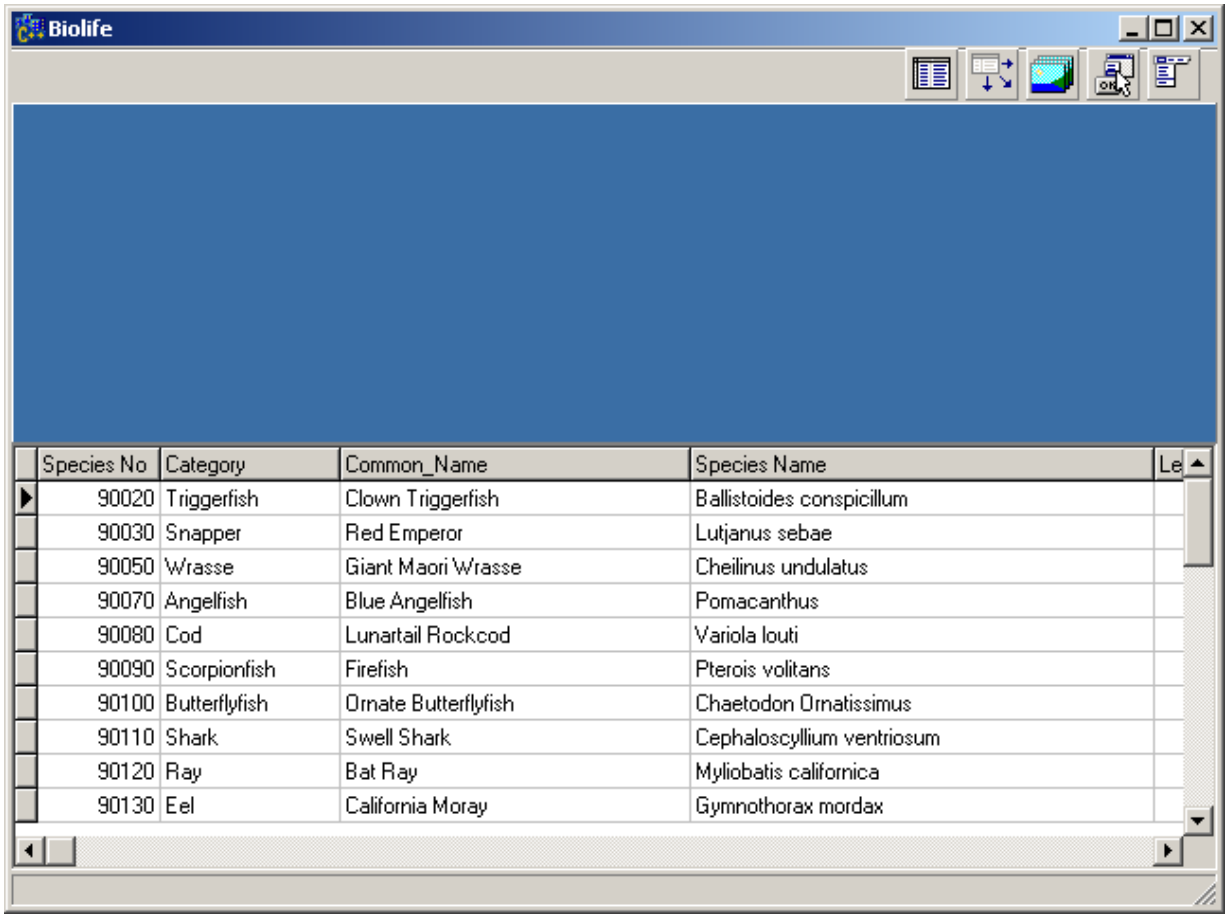


Рис. 7.2. Налаштування властивостей компонентів баз даних проекту 7.1

Активуйте компонент **Panel1**. Помістіть на панель компоненти **DBImage (**укладка **Data Controls), DBMemo (**укладка **Data Controls)** та **DBText (**укладка **Data Controls).** Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| DBImage1 | DBImage1 | Align | alRight |
| | | Width | 310 |
| | | DataSource | DataSource1 |
| | | DataField | Graphic |
| DBMemo1 | DBMemo1 | Height | 150 |
| | | Width | 310 |
| | | DataSource | DataSource1 |
| | | DataField | Notes |
| | | ScrollBars | ssVertical |
| DBText1 | DBText1 | Height | 25 |
| | | Width | 310 |

| | | | |
|---|---|---|---|
| | | Alignment | taCenter |
| | | DataSource | DataSource1 |
| | | DataField | Common_Name |
| | | Font->Style | fsBold=true |
| | | Font->Color | clSilver |
| | | Font->Size | 14 |

Збережіть проект.

Форма матиме вигляд як на рис. 7.3:

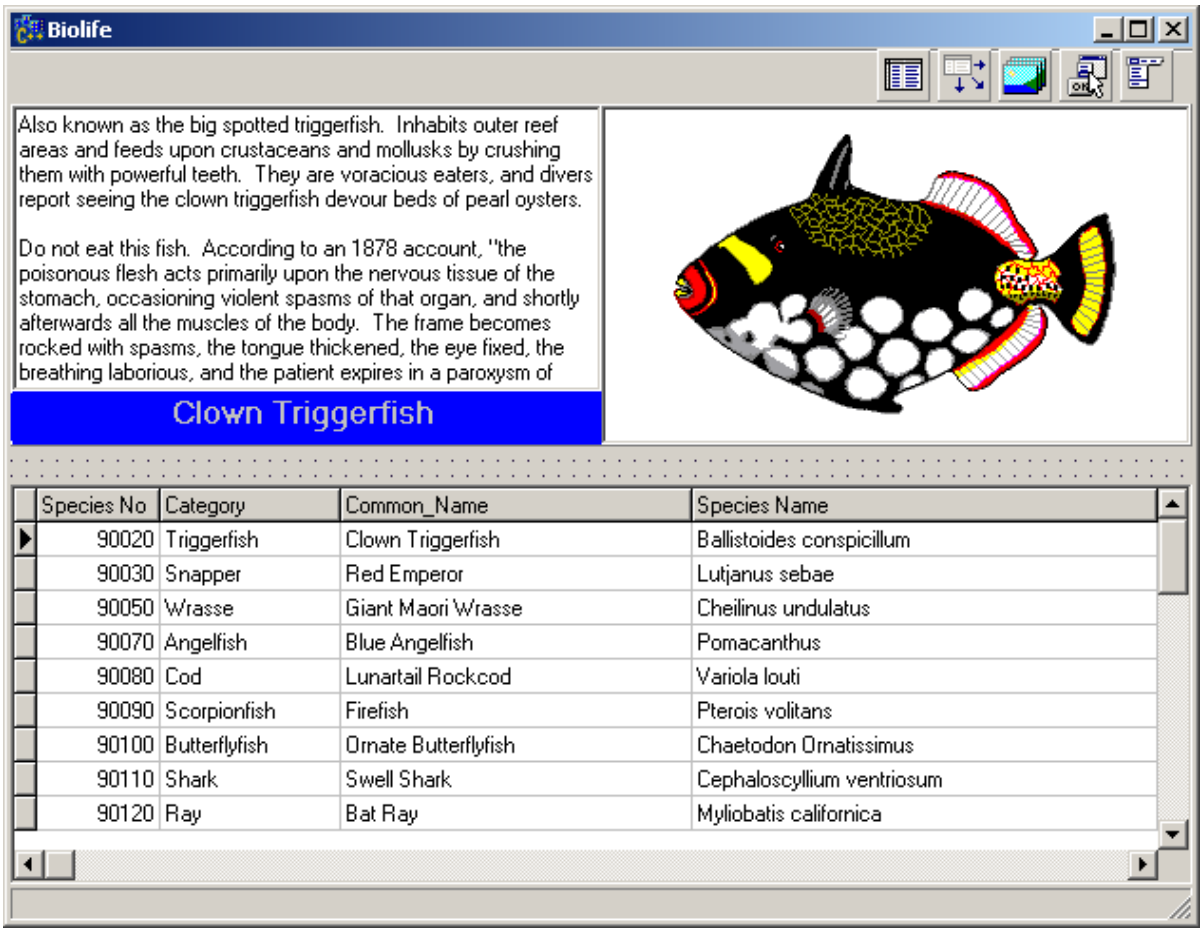


Рис. 7.3. Вікно програмного додатку проекту 7.1

Методика розробки додатку передбачає виконання послідовності кроків:

1. *Створюємо список дій* додатку, які повинні бути доступні користувачеві через розділи меню та інструментальні панелі: **First**, **Prior**, **Next**, **Last**, **Cut**, **Copy**, **Paste**.

Викличте Редактор дій **Editing Form1->ActionList1.** Активуйте контекстне меню Редактора дій та оберіть команду **New Standard Action** (нова стандартна дія). У вікні **Standard Action Classes** в категорії **Edit** оберіть дії **TEditCut**, **TEditCopy** та **TEditPaste** (використовуйте клавішу Ctrl для вибору декількох дій одночасно), в категорії **DataSet** оберіть дії

**TDataSetFirst**, **TDataSetPrior, TDataSetNext, TDataSetLast** (рис. 7.4):

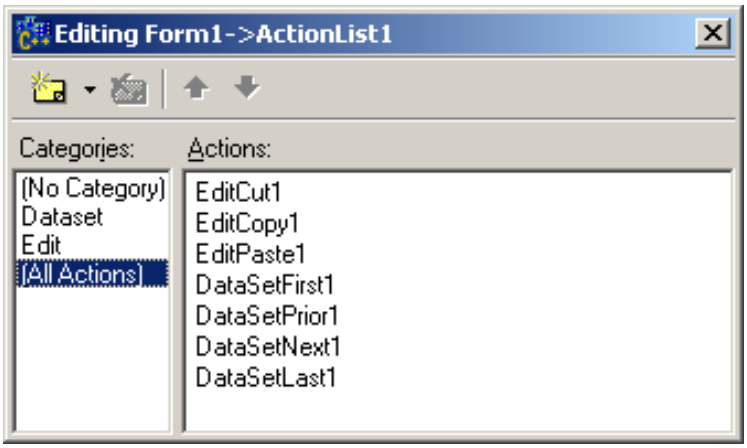


Рис. 7.4. Вікно редактору дій проекту 7.1

Встановіть значення їх властивостей згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| TEditCut1 | TEditCut1 | ImageIndex | 0 |
| TEditCopy1 | TEditCopy1 | ImageIndex | 1 |
| TEditPaste1 | TEditPaste1 | ImageIndex | 2 |
| TDataSetFirst1 | TDataSetFirst1 | ImageIndex | 3 |
| TDataSetPrior1 | TDataSetPrior1 | ImageIndex | 4 |
| TDataSetNext1 | TDataSetNext1 | ImageIndex | 5 |
| TDataSetLast1 | TDataSetLast1 | ImageIndex | 6 |

Закрийте Редактор дій.

2. *Заповнюємо список зображень* **ImageList1** піктограмами, які будуть використовуватися для тих дій, які повинні бути доступними зі швидких кнопок панелі інструментів (**Cut**, **Copy**, **Paste, First, Prior, Next, Last)**.

Подвійним клацанням миші на компоненті **ImageList1** відкрийте Редактор зображень **Form1->ImageList1 ImageList.** Клацніть на кнопці **Add** та перейдіть у директорію

**C:\Program Files\Common Files\Borland Shared\Images\Buttons**.

Додайте до списку зображень файли: **cut.bmp, copy.bmp**, **paste.bmp vcrfirst.bmp, vcrplayback.bmp, vcrplay.bmp та vcrlast.bmp** (рис. 7.5):

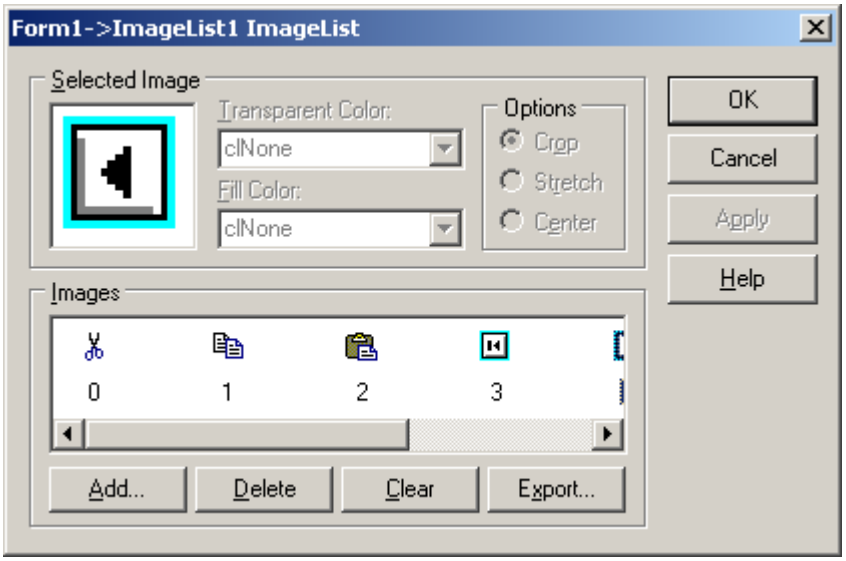


Рис. 7.5. Вікно редактору зображень проекту 7.1

Закрийте редактор **ImageList** кнопкою **OK**.

Перевірити список дій та відповідних зображень можна за допомогою Редактору дій **Editing Form1->ActionList1** (рис. 7.6):

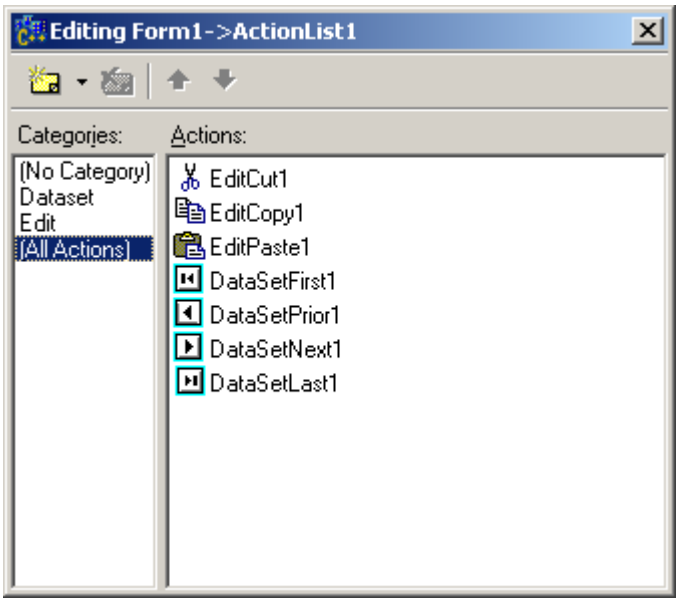


Рис. 7.6. Список дій і відповідних зображень проекту 7.1

Збережіть проект.

3. *Створюємо головне меню* додатку.

Подвійним клацанням миші на компоненті **MainMenu1** відкрийте Дизайнер меню **Form1->MainMenu1.**

Встановіть імена та властивості пунктів стандартного меню додатку згідно таблиці:

| **Компонент** | **Name** | **Caption** |
|---|---|---|
| TMenuItem | FileMenu | &File |
| TMenuItem | EditMenu | &Edit |

| | | |
|---|---|---|
| TMenuItem | RecordMenu | &Record |

Активуйте пункт меню **FileMenu**. Встановіть імена та властивості команд даного пункту меню згідно таблиці:

| **Компонент** | **Name** | **Caption** |
|---|---|---|
| TMenuItem | SaveFile | &Save |
| TMenuItem | ExitFile | E&xit |

Активуйте пункт меню **EditMenu**. Встановіть імена та властивості команд даного пункту меню згідно таблиці:

| **Компонент** | **Name** | **Caption** | **Action** |
|---|---|---|---|
| TMenuItem | CutEdit | Cu&t | EditCut1 |
| TMenuItem | CopyEdit | &Copy | EditCopy1 |
| TMenuItem | PasteEdit | &Paste | EditPaste1 |

Активуйте пункт меню **RecordMenu** (рис. 7.7). Встановіть імена та властивості команд даного пункту меню згідно таблиці:

| **Компонент** | **Name** | **Caption** | **Action** |
|---|---|---|---|
| TMenuItem | FirstRecord | F&irst | DataSetFirst1 |
| TMenuItem | PriorRecord | P&rior | DataSetPrior1 |
| TMenuItem | NextRecord | &Next | DataSetNext1 |
| TMenuItem | LastRecord | &Last | DataSetLast1 |

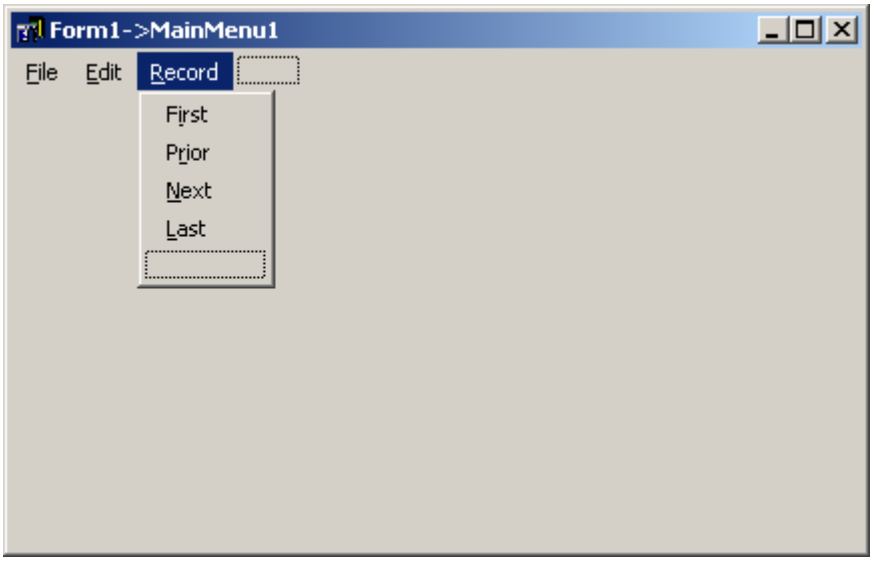


Рис. 7.7. Створення стандартного меню додатку проекту 7.1

Закрийте Дизайнер меню **Menu Designer Form1->MainMenu1.**

Збережіть проект.

4. *Створюємо панель інструментів* додатку.

За допомогою контекстного меню компонента **ToolBar1** додайте на панель три кнопки (**Cut, Copy, Paste**), роздільник та ще чотири кнопки (**First,**

**Prior, Next, Last**).

Встановіть імена та властивості кнопок згідно таблиці:

| **Компонент** | **Name** | **Action** |
|---|---|---|
| ToolButton1 | ToolButton1 | EditCut1 |
| ToolButton2 | ToolButton2 | EditCopy1 |
| ToolButton3 | ToolButton3 | EditPaste1 |
| ToolButton4 | ToolButton4 | |
| ToolButton5 | ToolButton5 | DataSetFirst1 |
| ToolButton6 | ToolButton6 | DataSetPrior1 |
| ToolButton7 | ToolButton7 | DataSetNext1 |
| ToolButton8 | ToolButton8 | DataSetLast1 |

Збережіть проект.

Форма матиме вигляд як на рис. 7.8:

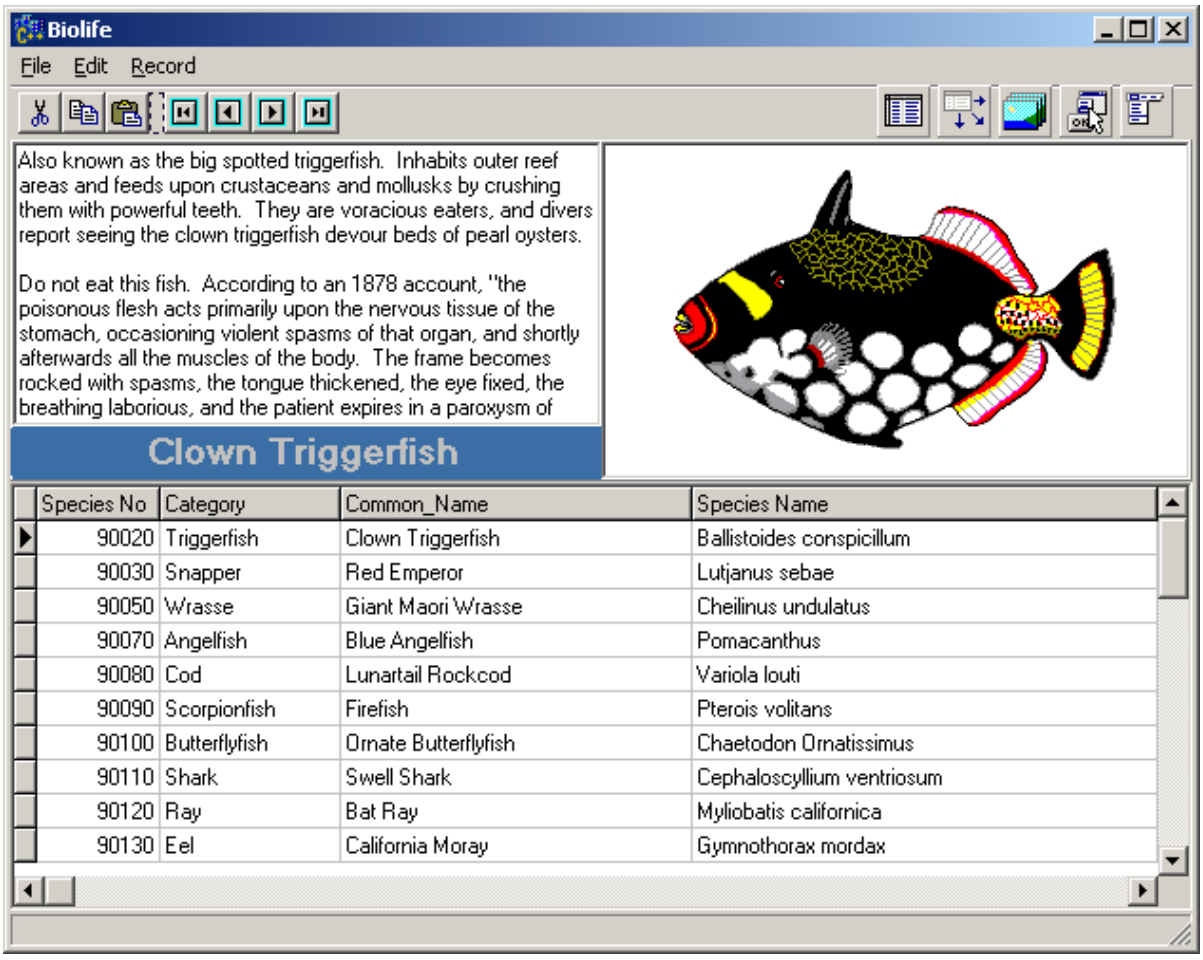


Рис. 7.8. Створення панелі інструментів додатку проекту 7.1

5. *Прописуємо обробники подій* для пункту **File** стандартного меню додатка.

Помістіть на форму компонент **SaveDialog** (укладка **Dialogs).**

Відкрийте Дизайнер меню **Form1->MainMenu1.**

Активуйте компонент **SaveFile** та створіть для нього обробник

події **OnClick**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::SaveFileClick(Tobject *Sender)
{
std::FILE *outfile;
AnsiString Buffer;
Buffer.sprintf("Save Info For: %s", DBText1->Field->AsString.c_str());
SaveDialog1->Title = Buffer;
if (SaveDialog1->Execute())
 {
  outfile = std::fopen(SaveDialog1->FileName.c_str(), "wt");
  if (outfile)
   {
    fprintf(outfile, "Facts on the %s\n\n", DBText1->Field->AsString.c_str());
     for (int i=0; i < DBGrid1->FieldCount; i++)
      fprintf(outfile, "%s: %s\n", DBGrid1->Fields[i]->FieldName.c_str(),
              DBGrid1->Fields[i]->AsString.c_str());
     fprintf(outfile, "\n%s\n", DBMemo1->Text.c_str());
   }
fclose(outfile);
 }
}
//---------------------------------------------------------------------------
```

Активуйте компонент **ExitFile** та створіть для нього обробник події **OnClick**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::ExitFileClick(Tobject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
```

**У** файл модуля **Baza.cpp** додайте директиву **#include <cstdio>.**

Збережіть проект.

Запустіть програму на виконання. Вікно проекту матиме вигляд як на рис. 7.9

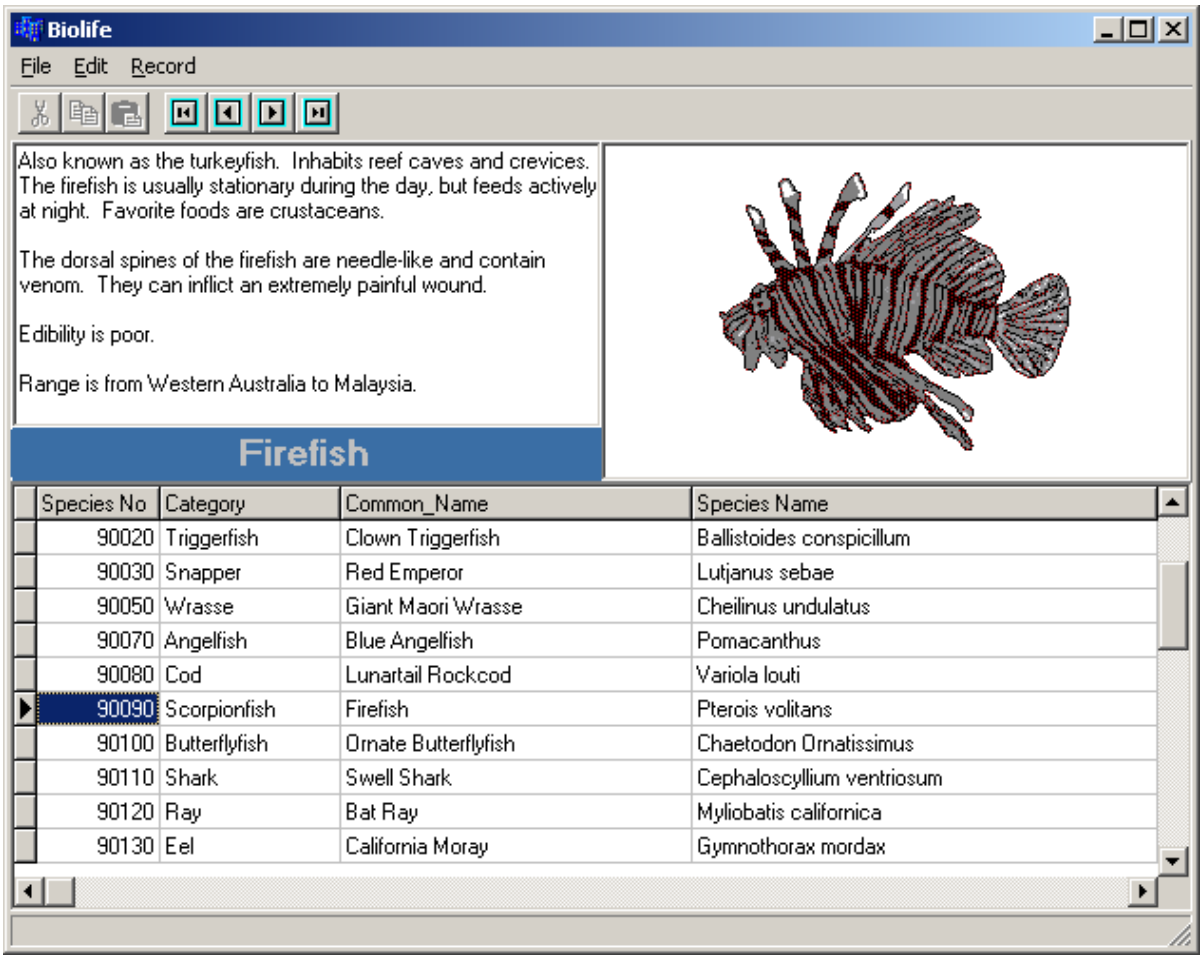


Рис. 7.9. Вікно програмного додатку проекту 7.1

Програма використовує стандартну базу данних biolife.db з набору C++Builder, яка постачається та встановлюється разом із C++Builder.

**Проект 7.2.**

Створіть на диску **E:** у своєму каталозі новий проект та збережіть його: модулю **Unit1.cpp** надайте ім'я **Baza_New.cpp**, файлу проекту **Project1.bpr** – ім'я **DB_New. bpr**. Крім того, в своєму каталозі створіть нову папку **DataBase** для зберігання таблиць бази даних.

Новий проект передбачає створення бази даних типу **Paradox,** в якій будуть зберігатися дані про ваших знайомих, а також створення програми керування цією базою даних.

Відкрийте **Database Desktop.**

Створіть псевдонім Вашої бази даних. Для цього у меню **Tools** Database Desktop виберіть команду **Alias Manager...** У діалоговому вікні **Alias Manager** клацніть на кнопці **New**. В поле **Database alias** введіть псевдонім створюваної БД — **StudentDB,** а в полі **Path** вкажіть шлях до своєї папки DataBase. В полі **Driver Type** виберіть драйвер STANDART для доступу до таблиць даних створюваної бази даних (рис. 7.10). Клацніть на кнопці Keep

New. Клацніть на кнопці OK.

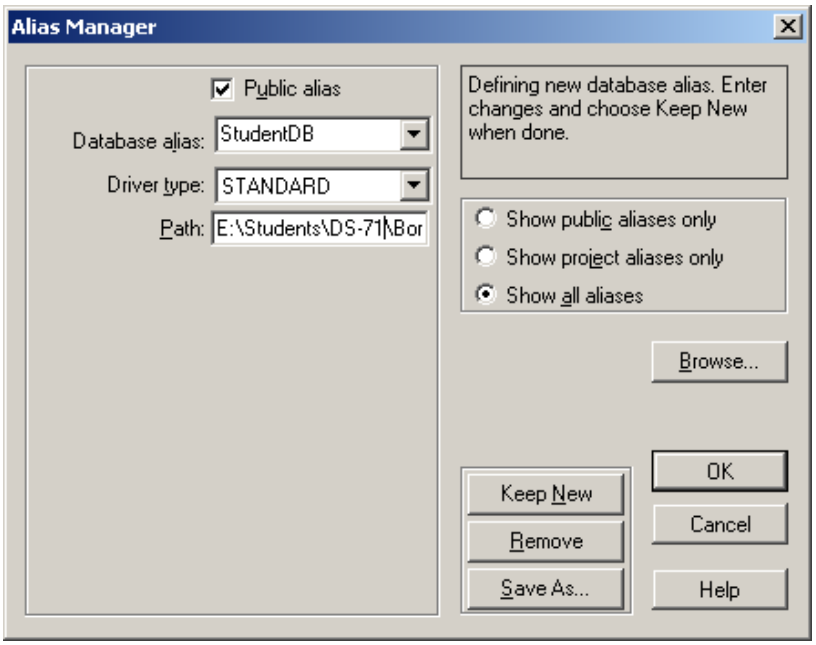


Рис. 7.10. Діалогове вікно Alias Manager

Створіть таблицю даних типу **Paradox 7.** Для цього у меню **File** Database Desktop виберіть команду **New->Table…** (рис. 2):

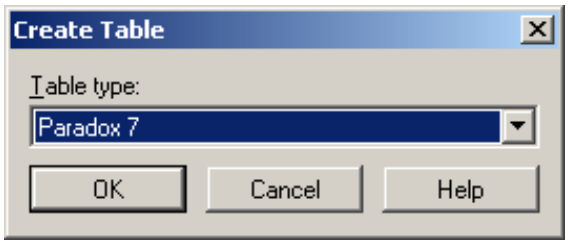


Рис. 7.11. Діалогове вікно Create Table

Встановіть імена та властивості полів даних таблиці відповідно до рис. 7.12:

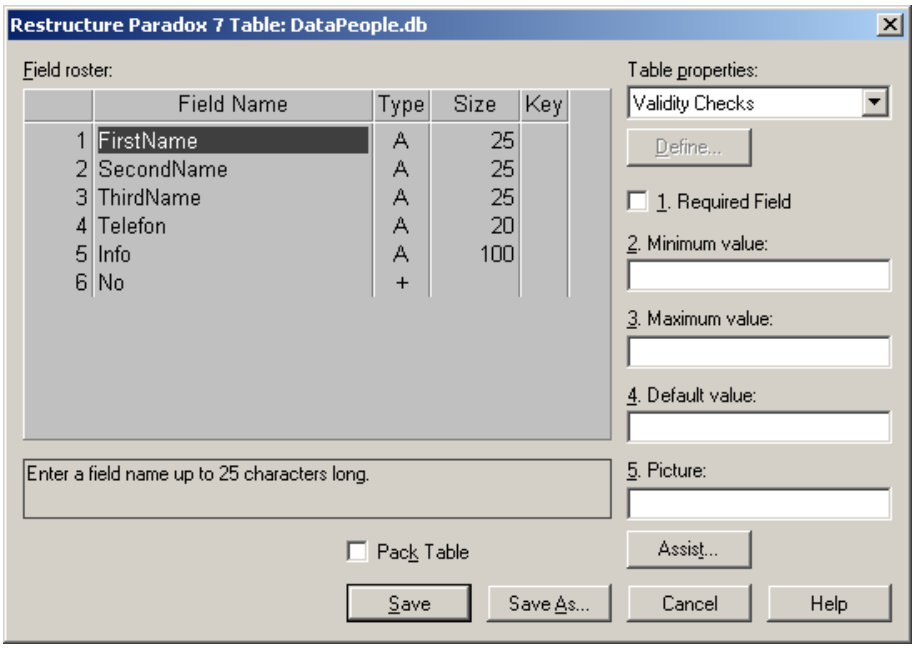

Рис. 7.12. Створення структури таблиці даних

Збережіть таблицю під іменем **DataPeople.db**. Внесіть декілька записів до таблиці. Збережіть зміни.

Помістіть на форму компонент **TabControl (**укладка Win32).

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Form1 | MainForm | Caption | UseDataDB |
| | | Height | 350 |
| | | Width | 630 |
| TabControl | TabControl1 | Align | alClient |

Помістіть на форму шість позначок **Label** (укладка **Standard**) та компоненти баз даних: **Table (**укладка **BDE), DataSource (**укладка **Data Access), DBMemo (**укладка **Data Controls)** й чотири компоненти **DBEdit (**укладка **Data Controls).** Розташування компонентів показано на рис. 7.13.

Встановіть імена та властивості компонентів згідно таблиці:

| **Компонент** | **Name** | **Властивість** | **Значення** |
|---|---|---|---|
| Label1 | Label1 | Caption | Особиста сторінка |
| | | Font->Name | Times New Roman |
| | | Font->Size | 20 |
| Label2 | Label2 | Caption | Прізвище |
| | | Font->Size | 14 |
| Label3 | Label3 | Caption | Ім’я |
| | | Font->Size | 14 |
| Label4 | Label4 | Caption | По-батькові |
| | | Font->Size | 14 |
| Label5 | Label5 | Caption | Телефон |
| | | Font->Size | 14 |
| Label6 | Label6 | Caption | Коментарі |
| | | Font->Size | 14 |
| Table1 | Table1 | DatabaseName | StudentDB |
| | | TableName | DataPeople.db |
| | | Active | true |
| DataSource1 | DataSource1 | DataSet | Table1 |
| DBEdit1 | DBEdit1 | DataSource | DataSource1 |

| | | | |
|---|---|---|---|
| | | DataField | FirstName |
| DBEdit2 | DBEdit2 | DataSource | DataSource1 |
| | | DataField | SecodName |
| DBEdit3 | DBEdit3 | DataSource | DataSource1 |
| | | DataField | ThirdName |
| DBEdit4 | DBEdit4 | DataSource | DataSource1 |
| | | DataField | Telefon |
| DBMemo1 | DBMemo1 | DataSource | DataSource1 |
| | | DataField | Info |

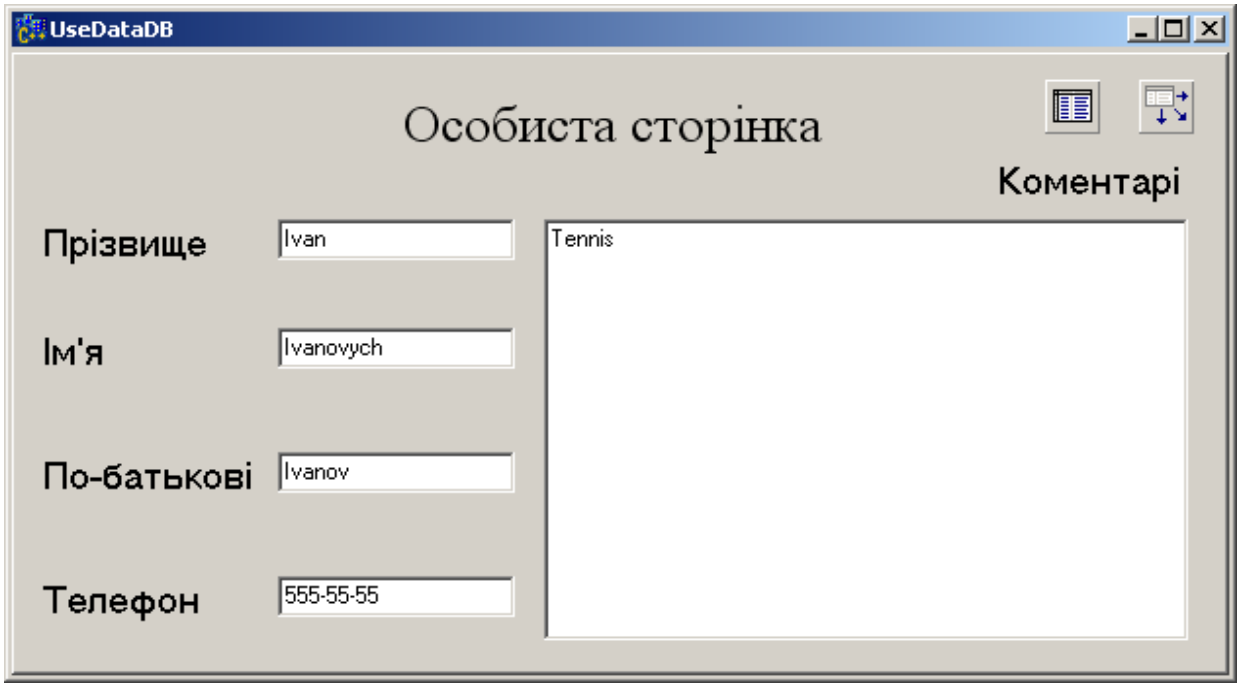


Рис. 7.13. Розташування компонентів проекту 7.2

Збережіть проект.

Активуйте форму **MainForm** та створіть для неї обробник події **OnCreate**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
 do
  {
   TabControl1->Tabs->Add(Table1->FieldByName("No")->AsString);
   Table1->Next();
  }
 while (!Table1->Eof) ;
Table1->First();
}
//---------------------------------------------------------------------------
```

Активуйте компонент **TabControl1** та створіть для нього обробник події **OnChange**:

```
//---------------------------------------------------------------------------
void __fastcall TMainForm::TabControl1Change(TObject *Sender)
{
 TLocateOptions Opts;
 Opts.Clear();
 Table1->Locate("No", TabControl1 -> Tabs ->
  Strings[TabControl1 -> TabIndex], Opts);
}
//---------------------------------------------------------------------------
```

Збережіть проект.

Запустіть програму на виконання. Вікно Вашого проекту матиме вигляд як на рис. 7.14:

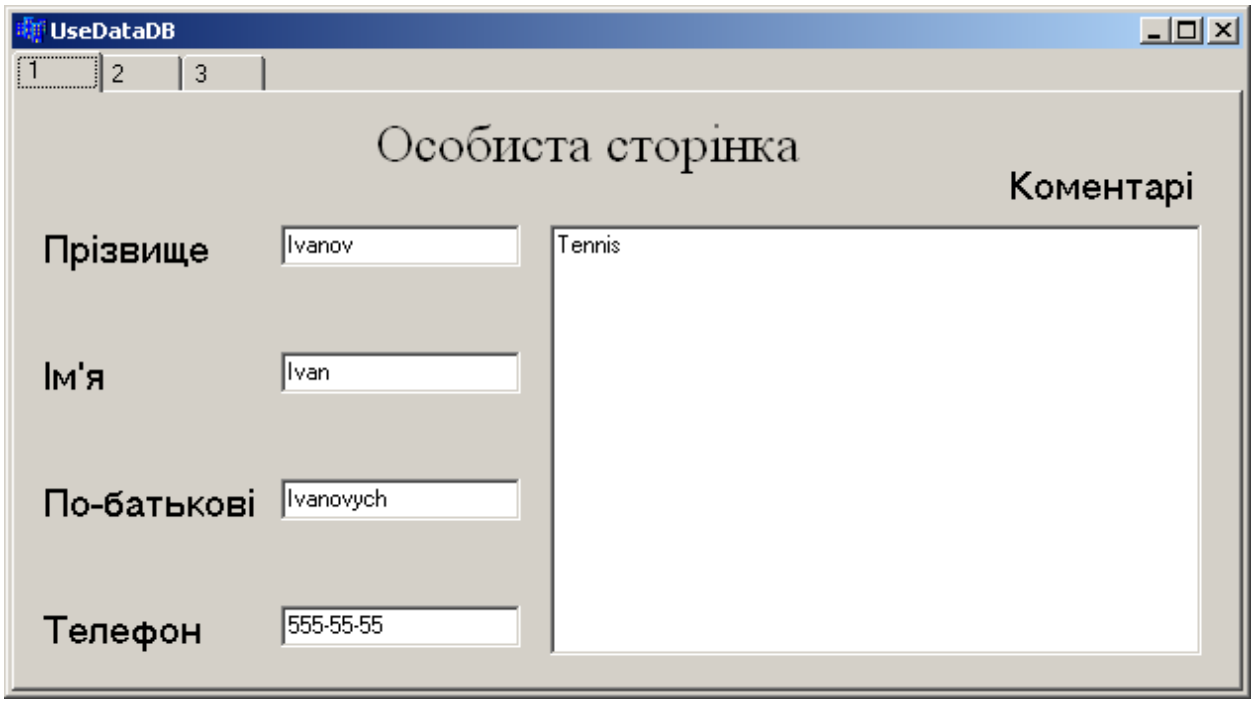


Рис. 7.14. Вікно програмного додатку проекту 7.2

**II. Контрольні завдання**

1. Додати можливість збереження інформації з бази даних.
2. Додати можливість сортування таблиці.
3. Додати список, в який би додавалися обрані записи БД.
4. Додати можливість збереження списку тварин у файл.
5. Додати можливість пошуку у базі даних.
6. Додати можливість вибору розміру зображення.
7. Додати можливість збереження зображення тварини.
8. Змініть проект таким чином, щоб в компоненті DBMemo виводилась

інформація з іншого поля БД.

9. Змініть проект таким чином, щоб в компоненті DBText виводилась інформація з іншого поля БД.
10. Змініть проект таким чином, щоб в компоненті DBGrid не виводилась інформація, яка відображається в компонентах DBText та DBMemo.
11. Змініть параметри шрифту окремого поля таблиці БД.
12. Додайте контекстне меню таблиці БД, через яке б змінювався розмір шрифту в таблиці.
13. Змініть код таким чином щоб замість цифр закладки підписувались прізвищами (імями , телефонами) записаних у книзі.
14. Додайте можливість пошуку по прізвищами (імями , телефонами) записаних у книзі.
15. Додайте кнопки за допомогою яких здійснюється навігація по книзі(вперед, назад)
16. Додайте список у якому будуть відображатись прізвища (імя , телефони) які за бажанням можна відсортувати.
17. Додати до бази даних графу яка буде зберігати фото( адресу, почтовий адрес, день народження, або інш) і реалізувати його відображення.
18. Додати ще один TabControl та реалізувати додавання до нього вибраних користувачем закладок.
19. Додати список і реалізувати додавання до нього вибраних користувачем прізвищ (імен , телефонів)
20. Додати можливість видалення, змінення або видалення запису із книги.
21. Додати можливість зберігання запису у файл.
22. Додати можливість завантаження запису з файлу.
23. Додати можливість додавання запису до книги.
24. Додати можливість зберігання всієї бази у текстовому форматі.
25. Відобразити усю телефонну книгу у таблиці.

### III. Контрольні питання

1. Назвіть компоненти доступу до бази даних.
2. Що таке псевдонім бази даних?
3. Які типи даних використовуються для таблиць типу **Paradox**?
4. Який компонент забезпечує доступ до таблиць бази даних?
5. Як заборонити редагування бази даних?
6. Які компоненти забезпечують відображення й редагування полів записів бази даних?
7. Назвіть основні властивості компонента **Table**.
8. Яке призначення компонента **DataSource**?

9. Назвіть основні властивості компонента **DateSource**.
10. Назвіть основні властивості компонента **DBGrid**.

## IV. Рекомендована література


1. Культин Н.Б. Самоучитель C++Builder. 2-е издание. / Н.Б. Культин – СПб.: БХВ-Петербург, 2008. – С. 134-163.
2. Послед Б.С. Borland C++Builder 6. Разработка приложений баз данных / Б.С. Послед – СПб.: ООО «ДиаСофтЮП», 2003. – 320 с.
3. Рейсдорф К. Borland C++Builder. Освой самостоятельно / Кент Рейсдорф, Кен Хендерсон – М.: «Издательство БИНОМ», 1998. – С. 529-597.
4. Сван Т. Основи программирования в Delphi для Windows 95 / Т. Сван. – К. : «Диалектика», 1996. – С. 386-406.

## СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ


1. Архангельский А.Я. Программирование в C++Builder / А.Я. Архангельский – М.: «Издательство БИНОМ», 2003. – 1152 с.: ил. – Библиогр.: с. 1150 – 4 000 экз. – ISBN 5-7989-0239-0.
2. Архангельский А.Я. C++Builder 6. Справочное пособие. Книга 2. Классы и компоненты / А.Я. Архангельский – М.: Бином-Пресс, 2002.– 528 с.: ил. – Библиогр.: с. 525 – 4 000 экз. – ISBN 5-9518-0009-9.
3. Боровский А.Н. C++Builder. Самоучитель. / А. Н. Боровский – Спб.: Питер, 2005. – 256 с.: ил. – 4 000 экз. – ISBN 5-469-00551-8.
4. Бобровский С. Самоучитель программирования на языке C++ в системе Borland C++Builder 5.0 / С. Бобровский – М.: ООО «ДЕСС КОМ», 2001. – 272 с. – 1 000 экз. – ISBN 5-93650-013-6, 5-9365-0013-6.
5. Глинський Я.М., Анохін В.Є., Ряжська В.А. C++ і C++Builder. Навч. посібн. 3-тє вид. – Львів: СПД Глинський, 2006. – 192 с.
6. Культин Н.Б. Самоучитель C++Builder. 2-е издание. / Н.Б. Культин – СПб.: БХВ-Петербург, 2008. – 464 с.: ил. – Библиогр.: с. 460 – 3 000 экз. – ISBN 978-5-9775-0268-9.
7. Культин Н.Б. C++Builder в задачах и примерах / Н.Б. Культин – СПб.: БХВ-Петербург, 2005. – 336 с.: ил. – 5 000 экз. – ISBN 5-94157-631-5.
8. Послед Б.С. Borland C++Builder 6. Разработка приложений баз данных / Б.С. Послед – СПб.: ООО «ДиаСофтЮП», 2003. – 320 с. – Библиогр.: с. 308 – 2 000 экз. – ISBN 5-93772-094-6.
9. Рейсдорф К. Borland C++Builder. Освой самостоятельно / Кент Рейсдорф, Кен Хендерсон – М.: «Издательство БИНОМ», 1998. – 704 с.: ил. – 6 000 экз. – ISBN 5-7989-0099-1.
10. Сван Т. Основи программирования в Delphi для Windows 95 / Т. Сван. – К. : «Диалектика», 1996. –480 с., ил.
11. Фейсон Т. Объектно-ориентированное программирование на Borland C++ 4.5 – К.: «Диалектика», 1996. – 544 с.: ил. – 6 000 экз. – ISBN 0-672-30605-0, 5-7707-9255-8.
12. Холингвэрт Д., Баттерфилд Д., Сворт Б., Оллсоп Д. C++ Builder 5. Руководство разработчика в 2-х томах / Холингвэрт, Баттерфилд, Сворт, Оллсоп – М.: Вильямс, 2001. – 880 с. – 2 000 экз. – ISBN 0-672-

31972-1, 5-8459-0201-0.

**ДЛЯ НОТАТОК**

**ДЛЯ НОТАТОК**

**ДЛЯ НОТАТОК**